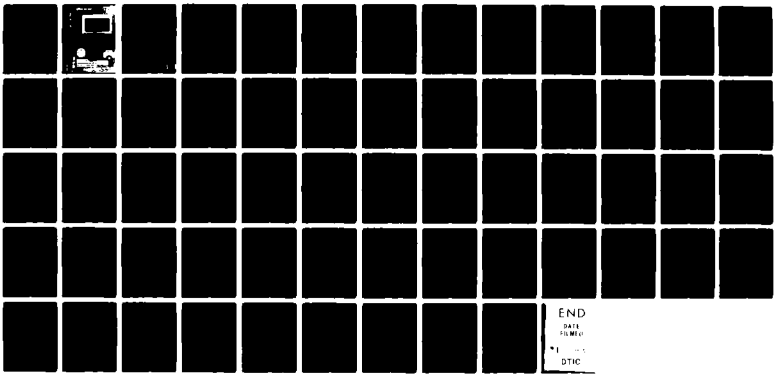
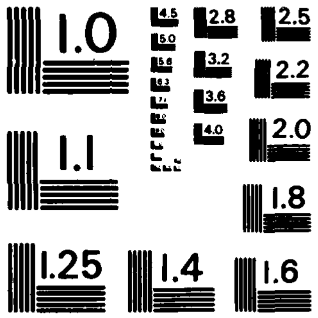
AD-A133 970 COMPUTATIONAL METHODS FOR COMPLEX FLOWFIELDS(U) 1/1
MASSACHUSETTS INST OF TECH CAMBRIDGE COMPUTATIONAL
FLUID DYNAMICS LAB E M MURMAN ET AL. 01 JUL 83
UNCLASSIFIED AFOSR-TR-83-0841 AFOSR-82-0136 F/G 12/1 NL



END
DATE
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

1983 ANNUAL REPORT

COMPUTATIONAL METHODS FOR COMPLEX FLOWFIELDS

Earll M. Murman
Judson R. Baron
Principal Investigators

AFOSR Grant 82-0136
OSP 92113

July 1, 1983

Computational Fluid Dynamics Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0841	2. GOVT ACCESSION NO. AD-A133 970	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Computational Methods for Complex Flowfields		5. TYPE OF REPORT & PERIOD COVERED Annual Report June 1, 1982 - May 31, 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Earll M. Murman, Judson R. Baron		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0136
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2307/A1
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NA Bolling AFB DC 20332		12. REPORT DATE July 1, 1983
		13. NUMBER OF PAGES 57
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Euler equations
Embedded grids
Adaptive grids
Airfoils
Computational Fluid Dynamics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The overall objective of this research is the development of solution algorithms for complex flowfields using procedures that take account of, recognize, and couple interacting subdomains. In Task I, non-adaptive embedded mesh solutions of the Euler equations have been obtained for NACA 0012, RAE 2822, and KORN 1 airfoils. In Task II adaptive embedded mesh solutions have been obtained for one-dimensional flows with shock waves and a two-dimensional scalar convection-diffusion equation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

1. Introduction -- 1
2. Research Objectives and Tasks -- 1
3. Summary of Task 1 - Nonadaptive Embedded Subdomains ------------------ 2
4. Summary of Task II - Adaptive Embedded Subdomains ------------------- 3
5. Cumulative List of Publications -------------------------------------- 5
6. Professional Personnel Associated with Research Effort --------------- 5
7. Interactions -- 5
8. New Discoveries, Inventions, etc. ------------------------------------ 5

Appendix A - Embedded Mesh Solutions of the Euler Equation Using a Multiple-Grid Method

Appendix B - Adaptive Embedded Subdomains

1. Introduction

Computational Fluid Dynamics (CFD) is capable of treating realistically complex aerodynamic problems. Prior to the mid-70's the main emphasis of CFD research was the development of numerical methods for solving the various governing equations of fluid dynamics. These include the potential flow approximation, Euler equations and Navier-Stokes. As 1980 approached, generalized coordinate transformations were developed which led to application of these algorithms to arbitrary geometries. The practical limitations for treating such geometries lie in the ability to generate a suitable grid, computer storage and computer speed capabilities. A reasonable approach to consider is a modular one wherein a general flowfield is considered to consist of a global scale and one or many local subdomains. In the subdomains more accurate grid resolution and/or equations may be required to resolve flowfield details. The issues to address then become how such calculations may be done in an accurate, efficient and manageable way.

2. Research Objectives and Tasks

The overall objective of the research sponsored by this grant is the development of solution algorithms for complex flowfields using procedures that take account of, recognize, and couple interacting subdomains. This objective recognizes that a discrete mathematics approach makes it possible to include, alter, and/or discard physical and geometric descriptions on a local basis, and that frequently no purpose is served in retaining an inefficient "complete" description. The local scale phenomena should then provide equivalently accurate contributions to a global scale description.

The effort is divided into separate considerations of flowfields with non-adaptive or adaptive embedded subdomains. The research tasks are concerned with both the decisions as to the allowable, necessary and realistic subdivisions, and the implied changes that must be made to the usually uniform application of a computational algorithm across a global field. In each case the background working algorithm has been a conservative, finite volume, multilevel method (Ni - AIAA J. Vol 20, No. 11) applied to the compressible flow equations. Detailed descriptions of the research performed during the past year are given in Appendices A and B. However, we summarize in the next two sections the scope and principal findings of this research.

3. Summary of Task 1 - Nonadaptive Embedded Subdomains

The solution of the two-dimensional Euler equations for flows past airfoils was considered in this task. Although some preliminary investigations considered simplified geometries, the reported results are for the NACA 0012, the RAE 2822 and the KORN 1 airfoil geometries. A body conforming O-type grid has been used for all calculations to date. The first stage of the research considered the solution of the flowfield on a single global grid. This corresponds to the traditional approach. The second stage of the research then considered one or more embedded subdomains. The subdomains have the same topology as the global domains, but a finer resolution. There are many significant findings which are discussed in detail in Appendix A and are summarized here. For the single global grid studies we found

. The computed value of the lift coefficient is strongly influenced by the far field boundary condition when the location of the outer boundary is at five to ten chords. If the far field effect of a vortex is included, the lift is accurate to a few percent, while if it is omitted the lift is in error by 20%.

. Total pressure loss is a sensitive measure of computational error.

. A characteristics type treatment of solid wall and far field inflow and outflow boundaries was satisfactory.

. When the mesh skewness at the boundary was large, significant total pressure errors were observed.

. Refined meshes generally led to improved accuracy except at the trailing edge.

. Various treatments of the Kutta condition were tried. The easiest and most versatile approach is to use no special treatment. This finding is in agreement with previous results but is not yet well understood.

For the embedded mesh studies the significant findings are

. Embedded mesh calculations have been found to give the accuracy of a globally refined grid if the embedded mesh is positioned "where the action is."

. No errors or computational stability problems were encountered when the discrete equations at the interface were developed consistent with the basic global solver. This was not found to be a difficult problem.

. The rate of convergence of the calculations with embedded meshes was found to be essentially the same as the single global grid. In some instances it was even better. This is perhaps the most surprising result of the research.

. The structure of the computational data base may be effectively treated using a pointer system similar to a connectivity array of finite element methods. This replaces the traditional i,j subscript notation and well ordered arrays of finite difference methods. There is a storage penalty for this approach, but conservative estimates indicate a significant net reduction in storage using embedded meshes.

. The net reduction of computational work for a given accuracy is significant using the embedded meshes.

. Cases with shock waves crossing the embedded mesh boundaries have been computed without difficulty.

. The procedure as currently constructed is not easily vectorizable for supercomputers. There are, however, various approaches for vector application which will be considered during the next year.

4. Summary of Task II - Adaptive Embedded Subdomains

Traditional adaptive grid approaches consider a fixed number of grid lines in each coordinate direction and a redistribution scheme based on achieving a uniform local truncation error. The current research considers a fixed global description with increasingly refined embedded regions. This is a fundamentally different approach which has required a fresh consideration of adaptive grid or adaptive equation methods. The scope of work for this first year of research has been to consider the general approach of adaptive subdomains and to perform computations on some test problems. The results are fully described in Appendix B and are summarized here. In considering the general approach to embedded subdomains the following concepts have evolved.

. The concept of a feature is introduced as a region in the flowfield which delineates dominant physics that is significantly different in scale, strength or orientation from the surrounding flow (e.g. shocks, vortices, shear layers).

. A search of the flowfield using a suitable criteria generates feature points.

. Feature points need to be clustered or grouped so they may be treated as a subdomain.

. The pointer system describing the data base structure permits the adaptive subdomains to be effectively treated in the computations.

. If the feature has a significantly smaller scale than the global scale it may be considered to be collapsible. It then may be replaced by a jump condition (shock), a singularity (vortex) or an altered boundary condition (displacement thickness) for example.

. Embedded region edges do affect the rate at which multilevel schemes accelerate convergence.

. A good criteria for whether the embedded subdomain needs to be more accurately modeled is the convergence of global parameters (lift, drag, etc) with increasing refinement.

The above concepts have been implemented in several test calculations using the one-dimensional Euler equation for variable area ducts and a two-dimensional scalar convection-diffusion (Burgers) equation. Significant findings include:

. For the one-dimensional duct calculations, adaptive subdomains (or sub-intervals) adequately found and resolved shock features. Savings in computational time, including all the overhead of the adaptive algorithm, were found to be a factor of 2-3 for the test problem. Greater efficiencies can be anticipated for more interesting problems.

. For the two-dimensional problem a viscous shear layerlike feature oriented at 45° to the grid was adaptively resolved. Similar savings in computer time were realized. The importance of consistent treatment at the domain interfaces was observed.

. The work to date indicates that additional studies should be continued leading up to a realistic two-dimensional compressible flow problem.

5. Cumulative List of Publications

[1] Usab, W.J. and Murman, E.M., "Embedded Mesh Solutions of the Euler Equation Using A Multiple Grid Method," AIAA paper 83-1946 CP. July 1983. To be submitted to AIAA Journal.

[2] Usab, W.J., "Embedded Mesh Solutions of the Euler Equations Using A Multiple-Grid Method." MIT Doctoral Thesis (in preparation)

[3] Dannenhoffer, J. and Baron, J.R., "Adaptive Solution Procedure for Steady State Solution of Hyperbolic Equations." Submitted to AIAA Aerospace Sciences Meeting, January 1984.

6. Professional Personnel Associated with Research Effort

Principal Investigators

Earll M. Murman, Professor

Judson R. Baron, Professor

Graduate Students

William J. Usab, Jr., PhD candidate (degree expected Fall 1983)

John F. Dannenhoffer III, PhD candidate

7. Interactions

1. Papers, seminars, presentations

   None in reporting period

2. Interactions with DOD laboratories

   Jack Benek, CALSPAN Field Services, AEDC Division, Tullahoma, TN, received copy of abstract submitted for reference 1, section 5.

8. New Discoveries, Inventions, etc.

None

APPENDIX A

Embedded Mesh Solutions Of The Euler Equation Using A Multiple-grid Method

William J. Usab, Jr.
Earll M. Murman

AIAA 83-1946-CP July 1983

Presented at the 6th AIAA Computational Fluid Dynamics Conference

Danvers, Massachusetts
July 13-15, 1983

Computational Fluid Dynamics Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Ca bridge, MA 02139

AIAA 83-1946-CP

Embedded Mesh Solutions Of The Euler Equation Using A Multiple-grid Method

William J. Usab, Jr. *
Earll M. Murman **

Department of Aeronautics and Astronautics,
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

A computational procedure using a multiple-grid method with embedded mesh regions is developed for solving the two dimensional Euler equations. A pointer system is used to define the general multiple grid structure. The required boundary conditions and global/embedded interface conditions are described. Results are presented for two dimensional subsonic and transonic airfoils using embedded meshes to resolve flow details in the leading edge, trailing edge, and shock regions. The present method is shown to retain the global coarse mesh convergence rates while gaining the flow resolution in embedded regions of a correspondingly globally refined mesh. Through the use of embedded meshes the total storage and computational work is significantly reduced over that of a equivalent global refinement.

INTRODUCTION

Recent algorithm developments of Ni (ref. 1), Jameson, Schmidt, and Turkel (ref. 2), and Rizzi (ref. 3) have demonstrated that it is now possible to obtain solutions of the Euler equations for many problems with practical computing times. Extending these methods to more difficult flows and increasingly complex geometries reduces to a problem of generating optimal grid distributions with high grid resolution where required while minimizing unnecessary points and grid generated numerical error. Currently this grid problem is handled by developing better grid generators or, for very complex geometries, through the use of patching techniques (ref. 4). In the latter approach, the domain is subdivided into simple subdomains which are patched together through special boundary conditions (ref. 5).

An attractive approach to complex geometries is to recast the problem in the frame work of a multiple-grid structure. The multiple-grid structure consists of a relatively crude global grid covering the entire solution domain, and any number

----

* Research Assistant, Member AIAA.
** Professor, Associate Fellow AIAA.

of embedded local grids providing adequate resolution of local flow features. Furthermore, the discrete equations represented on this grid are solved as a simultaneous system rather than as patched regions with interfacing boundary conditions. In the present work, this is accomplished using Ni's multiple-grid algorithm. Such an approach has been suggested by Brandt (ref. 6) for elliptic type equations solved by multigrid methods. It was implemented by Brown (ref. 7) for the transonic potential equation. However the multiple-grid structure given here presents the first step towards a completely general modular approach to solving the complete compressible flow problems. In this modular approach any number of local subdomains might be used where higher resolution, local grid systems, or special equation sets are needed. A modular approach of this type has the advantage of being easily adaptable from one problem to another and also computationally efficient for steady state calculations if advantage is taken of the multiple-grid algorithm for accelerated convergence.

This paper represents the initial step towards this goal through solutions of the two dimensional Euler equations on a multiple-grid structure with one or more embedded mesh regions. A more detailed discussion of the results presented herein may be found in reference 8. The solution algorithm is an extension of the multiple-grid scheme for the Euler equations presented by Ni (ref. 1). This multi-grid type algorithm lends itself to the multiple-grid structure suggested above. Storage is kept to a minimum since the required information is only stored once for each mesh point on the finest level. In order to extend this scheme to completely general grid structures the solution algorithm must be separated from the grid structure. That is, the organization of the computational data base, comprised of the variables at node points, must not be determined by the solution algorithm. This has been accomplished through the development of a pointer system which defines the grid structure. The usual subscripted index notation (i,j) of finite difference procedures is replaced by a single numerical subscript to identify mesh points. The pointer system is very similar to the connectivity array which is used to define general finite element systems. Boundary conditions and their location, which also vary from problem to problem, must likewise be defined in this pointer system. With the grid-structure defined through a pointer system, a general solver may now be written in terms of these pointers. This separation of grid structure from the solver is the key to creation of a general modular approach to problems.

The present paper is concerned with the solution of the Euler equations. Johnson (ref. 9) has demonstrated that Ni's multiple-grid accelerator is easily extendable to the Reynolds averaged Navier-Stokes equation. The present authors, in an unpublished pilot study, also drew the same conclusion. It is felt that the multiple-grid structure method given herein should prove to be an attractive algorithm for Navier-Stokes calculations.

In the sections which follow, the governing equations will be defined followed by review of the basic Ni scheme. The conditions used at farfield and solid wall boundaries are described together with the Kutta condition. The extension to embedded mesh regions is then made through formulation of proper cell integrations at embedded mesh boundaries. Finally, with the general solver formulated, the pointer system which directs the solver is presented.

The results demonstrate the multiple-grid structure method and its flexibility for several two dimensional subsonic and transonic airfoil problems. In these cases, embedded mesh regions have been used to resolve flow details in the leading edge, trailing edge and shock regions. In order to achieve equivalent resolution using other methods, global grid refinement would be required leading to substantially larger computer storage requirements and lengthier calculations.

GOVERNING EQUATIONS

The two dimensional Euler equations for inviscid flow may be written in conservation form for a cartesian coordinate system as

$$U_t + F_x + G_y = 0 \tag{1a}$$

where

$$U = \begin{vmatrix} \rho \\ \rho u \\ \rho v \\ e \end{vmatrix} \qquad F = \begin{vmatrix} \rho u \\ \rho uu + p \\ \rho uv \\ \rho u h_o \end{vmatrix} \qquad G = \begin{vmatrix} \rho v \\ \rho uv \\ \rho vv + p \\ \rho v h_o \end{vmatrix} \tag{1b}$$

in terms of density ρ, cartesian (x,y) velocity components (u,v), and total internal energy per unit volume e. The pressure p and total enthalpy h_o are then defined for a perfect gas as

$$p = (\gamma - 1)[\, e - 0.5\,\rho(\, uu + vv\,)] \tag{2}$$

$$h_o = (\, e + p\,)/\rho$$

where γ is the ratio of specific heats.

By use of the divergence theorem the governing equation may be written in integral form as

$$\frac{\partial}{\partial t}\iint_V U\, dA = \oint_{\partial V}(F,G).\vec{n}\, dS \tag{3}$$

Approximation of this equation leads to a finite volume method in conservation form.

BASIC MULTIPLE-GRID METHOD

The Ni multiple-grid algorithm is composed of two parts. The first part is a single step explicit Lax-Wendroff type time marching solver which is used on the solution mesh. To illustrate the basic Ni algorithm, the solution mesh is considered to be comprised of a single global grid called the level h mesh. The second part is a coarse mesh accelerator which operates on residuals transferred from the solution mesh to one or more progressively coarser grids. The key to both parts of Ni's multiple-grid scheme is the formulation of the discrete equations in terms of a control volume integration of the governing equations over each grid cell. The sum of this control volume integration, which may be called the cell residual or change, is then transferred to the surrounding grid points by way of a "distribution" formula. The resulting formulae for the corrections to grid point variables is equivalent to a standard Lax-Wendroff time step at each grid point.

It is not the intent of the paper to rederive the Ni formulation which is clearly presented in ref. 1. Rather, for completeness, the final formulation of the multiple-grid method will be briefly outlined for a general nonorthogonal grid system. In the present presentation, both the basic solver and the coarse mesh accelerator will be expressed in a cell reference frame using numerical values for grid points, cell centers, etc.. The reason for this choice of reference frame will become clear with the description of the pointer system in a later section.

The fine mesh solver begins with the initialization of all grid point corrections ($\delta U = U^{n+1} - U^{n}$) to zero. Then, cell by cell the following control volume flux balance and distribution are performed for each cell. For the typical cell shown in figure 1 this involves the following 4 steps.

STEP 1: Finite volume approximation

$$\Delta U_c = \text{Cell Residual}$$

$$= (\Delta t/\Delta V)\{ [0.5(F_1+F_2)(y_2-y_1)-0.5(G_1+G_2)(x_2-x_1)] \tag{4a}$$
$$-[0.5(F_3+F_4)(y_3-y_4)-0.5(G_3+G_4)(x_3-x_4)]$$
$$+[0.5(G_1+G_4)(x_4-x_1)-0.5(F_1+F_4)(y_4-y_1)]$$
$$-[0.5(G_2+G_3)(x_3-x_2)-0.5(F_2+F_3)(y_3-y_2)]\}$$

where

$$\Delta V = -0.5[(x_3-x_1)(y_4-y_2) - (x_4-x_2)(y_3-y_1)] \tag{4b}$$

This step is a discrete approximation to the governing integral equation on a cell volume whose shape is invariant with time.

STEP 2: Distribution formulae

$$\delta U_1 = \delta U_1 + 0.25[\Delta U_c - \Delta f_c - \Delta g_c] \tag{5a}$$
$$\delta U_2 = \delta U_2 + 0.25[\Delta U_c - \Delta f_c + \Delta g_c]$$
$$\delta U_3 = \delta U_3 + 0.25[\Delta U_c + \Delta f_c + \Delta g_c]$$
$$\delta U_4 = \delta U_4 + 0.25[\Delta U_c + \Delta f_c - \Delta g_c]$$

where

$$\Delta f_c = (\Delta t/ \Delta V)[\Delta F_c \Delta y^1 - \Delta G_c \Delta x^1] \tag{5b}$$

$$\Delta g_c = (\Delta t/ \Delta V)[\Delta G_c \Delta x^m - \Delta F_c \Delta y^m]$$

and

$$\Delta F_c = (\partial F/ \partial U)_c \Delta U_c \tag{5c}$$

$$\Delta G_c = (\partial G/ \partial U)_c \Delta U_c$$

$$\Delta x^{l} = 0.25(x_2 + x_3 - x_1 - x_4)$$

$$\Delta y^{l} = 0.25(y_2 + y_3 - y_1 - y_4)$$

$$\Delta x^{m} = 0.25(x_3 + x_4 - x_1 - x_2)$$

$$\Delta y^{m} = 0.25(y_3 + y_4 - y_1 - y_2)$$

$$U_c = 0.25(U_1 + U_2 + U_3 + U_4)$$

This step "distributes" the cell residual of step 1 proportionally to the solution grid points resulting in a Lax-Wendroff type formulation of the grid point correction equations. Expressed in this form, the numerical signal propagation phenomena appears similar in nature to characteristics propagation (ref. 1). In this distribution formula $(\partial F/\partial U)_c$ and $(\partial G/\partial U)_c$ are the Jacobian matrices evaluated at the cell center in terms of U_c. As Ni points out, a significant number of operations can be saved if ΔF_c and ΔG_c are directly formulated in terms of ΔU_c and U_c before coding.

STEP 3: Smoothing formulation

$$\delta U_1 = \delta U_1 + 0.25\,\mu[U_c - U_1] \quad (6)$$

$$\delta U_2 = \delta U_2 + 0.25\,\mu[U_c - U_2]$$

$$\delta U_3 = \delta U_3 + 0.25\,\mu[U_c - U_3]$$

$$\delta U_4 = \delta U_4 + 0.25\,\mu[U_c - U_4]$$

$$\mu = \sigma \Delta t\,[\Delta l + \Delta m]/\Delta V$$

$$\Delta l = \sqrt{(\Delta x^{l})^2 + (\Delta y^{l})^2}$$

$$\Delta m = \sqrt{(\Delta x^{m})^2 + (\Delta y^{m})^2}$$

While Lax-Wendroff type algorithms are known to have a significant amount of implicit artificial smoothing, for transonic and supersonic flows with shocks additional explicit artificial smoothing is required to stabilize the solution. From the authors' experience, when the multiple-grid accelerator is used this smoothing greatly improves the convergence rate, and, in many cases, is required for convergence. The present smoothing formulation (ref. 1), expressed here in a distribution format, would in practice be included in the distribution of step 2. It is important to note that it is the nodal solution that is being smoothed, using a nine point Laplacian operator, and not a smoothing applied to the cell. This smoothing is equivalent to adding a term of order Δx to the original governing equations if $\Delta x = \Delta y$. For a cartesian system this term is

$$\sigma\Delta x \{ U_{xx} + U_{yy} \} \tag{7}$$

STEP 4: Boundary conditions and solution update

Once the distribution has been performed at each cell on the fine mesh the required boundary conditions are applied. These will be discussed in more detail later. At each grid point over the complete solution mesh the dependent variables are updated by

$$U_i^{n+1} = U_i^n + \delta U_i \tag{8}$$

The newly calculated value of U_i is equivalent to a second order accurate (in time) Lax-Wendroff method.

This completes the formulation of the basic solver on the solution mesh with exception of the definition of the time step restriction. For stability the following relation is required

$$\Delta t <= \mathrm{MIN} \left| \frac{\Delta V}{|u\Delta y_l - v\Delta x_l| + a\Delta l} , \frac{\Delta V}{|u\Delta y_m - v\Delta x_m| + a\Delta m} \right| \tag{9}$$

where a is the speed of sound.

The above expression is based on the maximum eigenvalues of the Jacobian matrices ($\partial F/\partial U$ and $\partial G/\partial U$) of the quasilinear form of the governing equations for a nonorthogonal coordinate system and the corresponding cell dimensions. This stability restriction, based on experimental observations of Ni and the present authors, states that the maximum CFL number in both nonorthogonal coordinate directions must be less than or equal 1. It is interesting to note that a Von Neumann analysis of the present algorithm applied to the two dimensional scalar wave equation predicts the more restrictive condition of

$CFL<=1/\sqrt{2}$, while analysis of the corresponding one dimensional algorithm yields the experimental result, $CFL<=1$.

If the basic solver is used without the coarse grid accelerator marching with a global time step based on the above relation yields second order time accurate solutions. However, if only steady state solutions are of interest, much faster convergence is possible if each cell is advanced at the local rather than global time step condition. Of course, if the multiple-grid accelerator is used then the solutions are no longer time accurate and local time stepping is also used.

The multiple-grid accelerator consists of application of the following procedure on one or more progressively coarser meshes. The process begins by elimination of every other grid line in both coordinate directions resulting in what will be called the 2h mesh. A typical coarse grid cell is shown in figure 2. Note we now have access to the next level finer grid points shown as points 5-9 in the figure. First δU_i at all 2h grid points is initialized to zero. Then, following the basic solver, the 2h mesh is swept cell by cell performing the following steps:

STEP 1: Finite volume approximation

To retain the accuracy of the level h mesh solution the change, or cell residual, for the center of the 2h cell is determined from a weighted average of the level h mesh corrections. The simplest form, and that used here, is straight injection of the fine grid corrections as

$$\Delta U_c^{2h} = \delta U_5^{h} \tag{10}$$

STEP 2: Distribution formulas

The distribution formulas for this step are the same as the previous STEP 2 except that the known solution at the cell center (i.e. node 5) is used for calculation of $(\partial F/\partial U_c)$ and $(\partial G/\partial U_c)$. In addition, no smoothing is done for the coarse mesh sweeps as is done in step 3 of the basic solver.

Once the above steps have been performed at all 2h cells, the boundary conditions are applied at all 2h boundary points. Then the corrections are interpolated back to the fine mesh using linear interpolation. Finally the boundary conditions are applied once again on the fine mesh and the solution is updated using equation 8.

The above coarse mesh accelerator is then repeated for progressively coarser meshes (i.e. 4h,8h,....). A complete multiple-grid cycle consists of one sweep through the level h solution mesh followed by a coarse 2h mesh sweep, followed by a

4h sweep, and so on to the coarsest mesh.

BOUNDARY CONDITIONS

Each of the boundary conditions used has been implemented in a predictor/corrector form. The predictor/corrector form follows from the fact that second order numerical integration schemes for internal points incorporate a mathematical signal propagation phenomena analagous to the theory of characteristics. For example, Abbett (ref. 10) and others have viewed McCormacks scheme as computing the solution of two simple waves, the solutions of which are summed to yield a complete solution. In the same sense Ni suggests that the "distribution" formula represent similar simple wave solutions. On boundaries the predictor step consists of summing contributions from cells interior to the boundary. The corrector step consists of enforcement of the appropriate boundary conditions (i.e. inflow, outflow, solid wall, or Kutta) using a simple wave type of treatment.

In this section, subscript "p" defines predicted values obtained by distributions from the two boundary cells belonging to point i. Subscript "c" refers to the corrected values after application of the boundary conditions. Once found the corrected change at boundary points is then

$$\delta U_i = U_c - U_i^n \tag{11}$$

The corrector step for the farfield and solid wall boundaries is based on a characteristic analysis of the linearized Euler equations in a coordinate system tangential and normal to the boundary at point 1, as shown in figure 3. A general and easy to follow development of this characteristic analysis is presented by McCartin (ref. 11). If q_n and q_t are defined as the normal and tangent velocity components, and a is the speed of sound, then the eigenvalues λ and corresponding characteristic variables W in the reference frame normal to the boundary are

$$\lambda = \begin{vmatrix} q_n \\ q_n \\ q_n + a \\ q_n - a \end{vmatrix} \qquad W = \begin{vmatrix} \rho - p/(\bar{a}^2) \\ q_t \\ [\, q_n + p/(\bar{\rho}\,\bar{a})]/\sqrt{2} \\ [-q_n + p/(\bar{\rho}\,\bar{a})]/\sqrt{2} \end{vmatrix} \tag{12}$$

Bar quantities are linearized state conditions which are taken as the predictor state (p).

The number of boundary conditions which may be specified at the boundary is equal to the number of positive eigenvalues. For the far field the specification of the boundary conditions depends upon whether the normal velocity is positive (inflow) or negative (outflow) and supersonic or subsonic. For subsonic inflow $(0<q_n<c)$ the three positive eigenvalues require three boundary conditions be applied while W4 must come from the p state. The boundary conditions are set by defining W1, W2, and W3 in terms of freestream conditions(∞). For the moment it will be assumed that these correspond to a uniform freestream flow projected normal and tangential to the boundary. We thus have the following system of equations defining the corrected state (c).

$$\rho_c - p_c/(\bar{a}^2) = \rho_\infty - p_\infty/(\bar{a}^2) \tag{13}$$

$$q_{t_c} = q_{t_\infty}$$

$$q_{n_c} + p_c/(\bar{\rho}\bar{a}) = q_{n_\infty} + p_\infty/(\bar{\rho}\bar{a})$$

$$-q_{n_c} + p_c/(\bar{\rho}\bar{a}) = -q_{n_p} + p_p/(\bar{\rho}\bar{a})$$

After recombination we have,

$$q_{t_c} = q_{t_\infty} \tag{14}$$

$$p_c = 0.5[\, p_\infty + p_p + \bar{\rho}\bar{a}(\, q_{n_\infty} - q_{n_p}\,)]$$

$$\rho_c = \rho_\infty + (\, p_c - p_\infty\,)/(\bar{a}^2)$$

$$q_{n_c} = q_{n_\infty} + (\, p_\infty - p_c\,)/(\bar{\rho}\bar{a})$$

For supersonic inflow $(q_n>c)$ all eigenvalues are positive and four boundary conditions are required. In this case the inflow boundary is frozen at the freestream conditions.

For subsonic outflow $(-c<q_n<0)$ there is only one positive eigenvalue and therefore one required boundary condition. On the outflow boundary the pressure is set at the freestream value, $p_c = p_\infty$. W1, W2, and W4 are determined from the predicted flow conditions. After rearranging we have the following relations,

$$p_c = p_\infty \qquad (15)$$

$$q_{t_c} = q_{t_p}$$

$$\rho_c = \rho_p + (p_\infty - p_p)/(\bar{a}^2)$$

$$q_{n_c} = q_{n_p} + (p_\infty - p_p)/(\bar{\rho}\bar{a})$$

For supersonic outflow ($q_n < -c$) all information comes from the predicted state p.

While the farfield boundary conditions as presented are very commonly used for lifting airfoil problems, this boundary must still be placed a large distance from the airfoil for good results. In practice, it is not uncommon to see a far field inflow radius on the order of 100 chords (ref. 12). This is the result of the fact that while there is a net circulation around lifting airfoils, the farfield condition assumes zero circulation. This problem has been greatly reduced for subsonic cases in the present calculations by adding to the freestream flow the farfield effect of a compressible point vortex centered at the airfoil. With this formulation the outer boundary can be placed much closer to the airfoil.

The farfield compressible potential for a vortex in a uniform flow has been derived by Ludford (ref. 13) as

$$\phi = q_\infty R \cos(\theta - \alpha) - (\Gamma/2\pi)\tan^{-1}[\beta \tan(\theta - \alpha)] \qquad (16a)$$

where

$$\beta = \sqrt{1 - M_\infty^2} \qquad (16b)$$

The circulation Γ is based on the lift coefficient found from a surface integration around the airfoil

$$\Gamma = 0.5\, q_\infty c\, C_L \qquad (17)$$

Based on the freestream quantities and the calculated lift coefficient the following vortex far field condition (v) is obtained:

$$q_{n_v} = q_{n_\infty} \tag{18}$$

$$q_{t_v} = q_{t_\infty} + q_\infty c\, C_L \beta / \{4\pi R[\cos^2(\theta - \alpha) + \beta^2 \sin^2(\theta - \alpha)]\}$$

$$p_v = \{p_\infty^{\frac{\gamma-1}{\gamma}} + (\gamma-1)\rho_\infty[\, q_\infty^2 - q_v^2\,]/(2\, p_\infty^{1/\gamma})\}^{\frac{\gamma}{\gamma-1}}$$

$$\rho_v = \rho_\infty\,(\,p_v\,/\,p_\infty\,)^{1/\gamma}$$

Now instead of the freestream conditions (∞) the new vortex conditions (v) are used in each of the far field inflow boundary conditions presented above. Although this formula strictly applies to irrotational flow, it has been used for flow involving shock wave generated vorticity.

Finite volume methods with the state vectors defined at cell centers (e.g. refs 2,3) only require the pressure on the solid wall. Incorporation of the solid wall condition for Ni's scheme requires all flow quantities be known or determined at the solid surface. For this reason, a characteristic analysis is also used at the solid walls. Referring back to the boundary cells shown in figure 3 and with q_n=0 in eqn 12, there is one positive eigenvalue λ_4 requiring one boundary condition be set. The condition used is q_n =0. W1, W2, and W3 are then determined based on the predicted state (p) where

$$(\,U_1\,)_p = U_1^n + 2(\,\delta\, U_1\,)_p \tag{19}$$

The factor of two in the above expression is used to accelerate convergence. This might be thought of as either a crude application of the reflection principle at the solid wall or merely an over-relaxation of the predicted change. After substitution and recombination the corrected conditions are found to be

$$q_{n_c} = 0 \tag{20}$$

$$q_{t_c} = q_{t_p}$$

$$p_c = p_p + q_{n_p}\,\bar{\rho}\bar{a}$$

$$\rho_c = \rho_p + q_{n_p}\,\bar{\rho}/\bar{a}$$

It is interesting to note that this solid wall boundary condition is a linearized version of the common simple wave boundary condition, where the corrected state is based on the generation of an isentropic expansion or compression wave normal to the boundary which is of sufficient strength to cancel q_n .

All airfoil solutions to be presented have been obtained on O-type meshes. This places a mesh point at the trailing edge of the airfoil which is a singular point in the flow field. The procedure used to enforce the body boundary condition should be modified to enforce a Kutta condition. A number of procedures were tried. The simplest and seemingly most reliable procedure to date is to use the predicted values from the distribution formula without applying any correction. In essence this amounts to doing nothing special at the trailing edge which agrees with published results (refs 2,3). The smoothing coefficient was increased for wall points in the trailing edge regions to eliminate minor oscillations in this region. It was verified for a subcritical case that this gave the same lift as specifying a flow direction along the bisecting angle of the trailing edge or specifying a condition which produced a stagnation point. Further discussion may be found in reference 8.

The smoothing formulation presented in step 3 of the fine mesh solver is not applied along the farfield and solid wall boundaries. Along these boundaries the smoothing component normal to the boundary is dropped for lack of information. The solution is smoothed tangent to the boundary using the corresponding one dimensional smoothing operator.

The basic multiple-grid Euler solver and boundary boundary conditions have been verified for several different flow problems. Figure 4 shows the near field of a 129*33 O-type mesh with a far field radius of 5 chords for a NACA0012 airfoil. This mesh, which will be defined as the h/2 mesh, was generated using a 2-D version of the transfinite interpolation routine described in ref. 14 and supplied to us by the author. The surface pressure coefficient, surface total pressure loss and near field Mach number contours are shown in Fig. 5 for M = 0.63 and a 2.0 degree angle of attack. A multiple-grid solution with 4 global mesh levels was used. The calculated lift coefficient of 0.329 agrees well with the theoretical value of 0.335 presented in Ref. 15. Total pressure loss, which should be zero, has been presented since it has been found to be a very sensitive indicator of errors in the boundary condition formulation and poor grid resolution (ref. 16). Based on this parameter the characteristic formulation of the solid wall boundary condition works quite well.

Figure 6 present the solution for the same flow conditions using a 65*17 mesh which corresponds to h mesh of the previous result. The only detectable changes from the 129*33 solution are the higher surface total pressure loss and the slightly lower lift coefficient of 0.327. These changes are the direct result of the poorer mesh resolution in the leading and trailing edge regions. Figure 7 compares the convergence histories of the two solutions. The average absolute value of the change $\delta(\rho u)/\Delta t$ on the finest mesh is plotted as a function of the multiple-grid cycles. Since both solutions were obtained by marching at the maximum local CFL condition, the large difference in convergence rates is due to the larger time step for the h mesh. If one includes the factor of four increase in work per multiple-grid cycle for the 129*33 mesh solution, the importance of minimizing the total number of grid points is clear.

The spike in the total pressure loss at the trailing edge for these cases is a localized effect due to skewness of the O-type mesh in this region. This error can be reduced, as shown in figure 8 for a 65*17 mesh, if the mesh is this region is made more orthogonal to the surface.

Each of the above calculations were performed using the vortex far field characteristic boundary condition. Table 1 presents the lift coefficients for solutions on a 65*17 mesh with a far field radius of 5 and 10 chords, with and without the far field vortex correction. If each is compared with the theoretical lift coefficient value of 0.335 it is clear that, while the errors in both cases drop off with increasing radius, by using the vortex correction the far field boundary may be brought much closer to the airfoil. This in turn reduces the storage and work by reducing the number of mesh points required for equivalent mesh resolution.

EXTENSION TO GENERAL EMBEDDED MESH FORMULATION

Consider the addition of a local embedded mesh of half the mesh spacing h/2 into the standard global mesh as shown in figure 9. After renumbering the mesh levels, h/2 being level 1, h being level 2, and so on, it is noted that level 2 is now a coarse mesh within the embedded region and a fine mesh outside this region. It is desired to perform a control volume flux balance for all cells on the finest mesh in each region of the total domain in order that the fine mesh accuracy be obtained. However, it is also desired to couple the solution of the discrete equations throughout the total domain in order to achieve rapid convergence. The solution begins on level 1, which includes only the embedded mesh region. Steps 1 to 4 of the basic solver are done for all cells in level 1, except those at the boundary which will be subsequently described. Proceeding to level 2, for those cells outside the embedded mesh region the basic fine mesh solver is used. However, for those cells within the embedded region the coarse mesh accelerator is used. For levels greater than level 2 it follows that the coarse mesh accelerator would be used everywhere. Beyond the basic framework just described, two special problems must be considered. One is the treatment of the boundary points and the other is the organization of the mesh points. The latter is described in the next section on the pointer system.

Points at the boundary must be carefully treated in order to maintain global conservation and computational stability. Consider the embedded mesh/global mesh interface shown in figure 10. A choice must be made as to whether points 1,6,2 are to be considered as members of the global grid or the embedded grid. That is, it must be decided as to whether the solution of the equations at these points is to be obtained to global or embedded grid accuracy. In this paper, the approach has been adopted that the boundary points are members of the global grid. The solution for points 1 and 2 is obtained on the level 2 sweep described above. Values at point 6, which are needed to compute the level 1 sweep, are obtained by linear interpolation from points 1 and 2. Linear interpolation is consistent with the trapezodial integration used on the flux balances.

Treatment of the boundary cells proceeds as follows. Prior to the solution sweep on level 1, points such as 6 are initialized by linear interpolation from points 1 and 2. Steps 1 through 3 of the basic fine mesh solver are performed for all cells on level 1 including those bounded by points 1,6,2. Prior to the execution of step 4, all values of δU at boundary points between the embedded and global mesh are reset to zero (points 1,6,2). Step 4 is then completed. As a result, no change of U has taken place at the boundary points. After deletion of every other point on the embedded mesh the level 2 solution outlined earlier proceeds except for boundary cells as shown in figure 10. The injected value from the fine mesh is used for coarse mesh accelerator updating of interior points such as 3 and 4. However a computed flux balance is used in the distribution formula for boundary points such as 1 and 2. It is obtained from the level 1 sweep and saved prior to execution of step 4. A trapezodial integration is done around the boundary of the cell with center point 5 using all the level 1 points as the boundary. It therefore has lower truncation error than level 2 cell integrations. Smoothing is added into the distribution formula for points 1,2. It should be noted that with the above formulation, smoothing is applied only once to each node point. Those within the embedded mesh are smoothed on the level 1 operations while those on the embedded/global mesh boundaries and the global mesh are smoothed on the level 2 operations.

To demonstrate the above embedded mesh formulation consider the NACA0012 test case of figures 4-7. Beginning with the h mesh (65*17) of figure 5 as the global mesh we now include embedded h/2 meshes around the leading and trailing edges of equal density to a 129*33 mesh as shown in figure 11. Using the embedded mesh formulation presented above produced the solution shown in figure 12. Comparing this solution with the 129*33 and 65*17 solutions (figures 5 and 6) shows that the embedded leading and trailing edge regions have resolved the same flow detail as the global 129*33 mesh (fig. 5). By using the leading edge embedded mesh the total pressure loss in this region (fig. 12b) is the same level as in figure 5b. A check of embedded mesh boundary regions has shown that no total pressure losses are generated in these regions due to the embedded mesh. A calculated lift coefficient for the embedded mesh solution of 0.330 is almost exactly the same as the global 129*33 mesh result.

The residuals presented for embedded mesh solutions are the average of the absolute value of $\delta(\rho u)/\Delta t$ for all points in the domain after the global h level sweep. The spectral radius for all other levels have been found to be the same. Figure 13 presents the the global h mesh residual convergence history showing solution convergence in nearly the same number of multiple grid cycles as the global 65*17 solution and almost half the number of cycles compared with the global 129*33 solution (fig. 7). Thus, we have gained the 129*33 mesh resolution with a convergence rate on the order of the 65*17 global solution. Since the total number of mesh points is much less than the number of global 129*33 mesh points, the work per cycle is also significantly reduced.

The extension of the above procedure to multiple embedded domains and embedded regions with more than one level follows directly. The solution cycle always begins of the finest mesh. Under the above described formulation, the boundary of a h/4 mesh embedded in a h/2 mesh should be at least a distance h from the boundary of the h and h/2 mesh. Some changes in the formulation could remove this restriction. However, this is not an important restriction since the truncation error near an interface will be of the coarser h level order anyway.

To illustrate the benefit of continued mesh refinement consider the NACA0012 embedded mesh structure of figure 11 but now include a second embedded h/4 mesh in the leading edge region as shown in figure 14. The solution for this double embedded region is shown in figure 15. As expected the higher resolution further reduced the total pressure loss and shows an improvement in the force coefficients when compared to figures 5,6, and 12. The convergence history on the global h mesh is similar to the previous embedded mesh case, with no loss in the rate of convergence with the addition of the new region embedded region.

POINTER SYSTEM

The purpose of the pointer system is to act as a kind of road map for the solver. The pointer system defines the multiple-grid structure including domains of each level of the structure, location of physical boundary conditions and embedded mesh boundaries. Separation of this structure definition from the Euler equation algorithm leaves a very general and easy to follow program. When the grid structure is changed, only the pointers are changed not the solver. With the present formulation the solution U and change U are stored only once for each grid point on the finest mesh in each local region. The pointer system then defines each grid level in terms of these basic nodes.

For the full two dimensional Euler equations the following 10 quantities must be stored for each node: coordinates x and y, conservation variables ρ, ρu, ρv, and e,and the change in the conservation variables $\delta\rho$, $\delta\rho u$, $\delta\rho v$, and δe. They are stored in a 10 by N solution matrix Q defined as

$$Q = [\, Q_{mn} \,] \tag{21a}$$

where

$$\begin{aligned} m &= \text{Variable Type} \quad (1 \le m \le 10) \\ n &= \text{Node Number} \quad (1 \le n \le N) \end{aligned} \tag{21b}$$

Additional quantities such as cell volumes, projected areas, temporary variables, etc. could also be stored to reduce repetitive calculations.

A cell pointer matrix must now be defined which points to the nodes in Q needed for the formulas on each level of the multiple-grid structure. There are many possible ways to define the grids of each level but the most basic is to use the smallest element, the cell. Other possible grid structures include contiguous lines or blocks of cells. However, defining the grid on each level in terms of cells allows the most flexibility in the definition of the grid structure. The domain of a given level does not need to be simply connected, topologically restricted, or even defined in any order. In addition, both the fine mesh solver and coarse mesh accelerators can be simply expressed in terms of cells as discussed earlier and shown in figures 1 and 2. Nine pointers are required to define the nine nodes of each cell of every level of the grid structure (figure 2). They are stored in an integer matrix P defined as

$$P = [P_{ij}] \tag{22a}$$

where

$$i = \text{Cell Node Number} \quad (1 <= i <= 9) \tag{22b}$$
$$j = \text{Cell Number} \quad (1 <= j <= J)$$

Note that for the fine mesh cells the injection and interpolation points (5<=i<=9) are set to zero since those nodes don't exist. The value of point 5 is then the "switch indicator" as to whether the fine solver or the coarse mesh accelerator should be applied.

For embedded mesh calculations nodes must be smoothed on different levels. This can be very efficiently handled by setting the sign of the corner pointers (1<=i<=4). If the node is to be smoothed by the cell it is positive, otherwise it is negative.

Finally there must be some way of knowing which level the cells belong to. By storing cells of the same level together (in any order), then only a pointer for the first and last cell of each level is required. The cell pointer matrix P combined with these level pointers completely defines the multiple-grid structure.

In addition to the basic grid structure, the location and type of boundary conditions must be defined for the solver. Boundary conditions are really exceptions to the general solver and can be problem dependent. Boundary conditions also tend to require different amounts of information and quite often access to domains larger than one cell. For these reasons, they are not included in the cell pointer matrix since once defined we would like this matrix definition to remain fixed.

At the present time there are two types of boundary pointers. As the need arises new forms can be added. Type 1 is used for solid wall, farfield boundaries, and any other boundary condition where pairs of cells are required. For each boundary node on the finest local mesh level the following information is stored in a 3 by K matrix called B1

$$B1 = [B1_{ik}] \tag{23a}$$

where

$i = 1$ Cell Number of Cell 1 (23b)
$\phantom{i = {}}2$ Cell Number of Cell 2
$\phantom{i = {}}3$ Face Number

k = Boundary Point Number

The four possible boundary locations are shown in figure 16. For more than one boundary condition defined by this pointer all points of the same type are stored together along with a starting and ending pointer for each boundary condition.

The second boundary pointer, type 2, is for boundary conditions that need only a cell side or string of 3 nodes as shown in fig 17. The embedded interface formulation is the only condition that uses this at this time. The definition of this pointer matrix B2 follows,

$$B2 = [\, B2_{ij} \,] \qquad (24a)$$

where

$i = 1$ For Node Number of Point 1 (24b)
$\phantom{i = {}}2$ For Node Number of Point 2
$\phantom{i = {}}3$ For Node Number of Point 3

j = Number of Interface Interpolation Point

This pointer is used to define the solution interpolation for point 2 along the embedded mesh interface before the embedded sweep and to zero the interface corrections at points 1,2, and 3 after the sweep. In B2 all sides on a given level are stored together from which a starting and ending pointer for each level are defined.

Clearly the pointer scheme described above provides a very flexible approach for dealing with complex grid structures. With an optimal grid structure the solution storage and computer time can be minimized. The price which must be paid for this flexibility appears in the total storage required and organization of the data base for vector computer architectures. While the solution storage is significantly reduced, the pointer system must also be stored. To illustrate the storage requirements let's compare the following two dimensional cases. In the first case we have a N*N global mesh with an embedded mesh (h/2) over one quarter of the domain as shown in figure 18a. For the second case we will consider a standard non-embedded mesh calculation where a global h/2 mesh refinement has been used to gain the same resolution as the first case, figure 18b. Storage of the solution in both cases requires storage of 10 real variables for each node of the finest mesh in each region (20 words(16 bit)/point);

(20)(1.75)*N*N words for the first case and (20)(4)*N*N in the second. Neglecting the boundary pointers, the pointer system requires 9 integer variables for each cell of each level (9 words(16 bit)/cell), assuming two grid levels, the total pointer storage for the embedded case is (9)(2)(N-1)(N-1) words. A summary of the storage requirements for the two cases is presented in table 2. Comparing the total storage there is a reduction by using the pointer system but this reduction is less than might be expected if the pointers were not required. However, if less than one quarter of the region used an embedded mesh or if two embedded meshes are used, or more than 10 variables are stored at each node point the ratios will change.

RESULTS

The embedded mesh extension to Ni's multiple-grid method as presented in the preceding sections provides a very flexible structure in which resolution of local flow detail is possible while minimizing the storage and work required. In general the addition of a local embedded h/2 level mesh region results in very little change in the number of multiple-grid cycles over the solution on the h global mesh alone. The price for this higher resolution then only appears in the additional work performed within the embedded mesh. The following three 2-D airfoil solutions demonstrate the performance and flexibility of the present formulation. Each of these cases has special flow details which must be resolved for a proper solution. They are often chosen for code comparisons (for example the first two were part of the GAMM workshop ref. 17). In each of the following solutions the far field vortex correction has been used with a mesh far field radius of 5 chords. Two solutions will be presented for each case the first being a 65*17 global mesh solution using 3 multi-grid levels and the second the corresponding embedded mesh solution using a total of 4 multi-grid levels.

The first case is a NACA0012 airfoil at flow conditions of $M_\infty = 0.85$ and $\alpha = 1.0$ deg.. This is a lifting case with strong shocks at 85% chord on the upper surface and 70% chord on the lower surface. The lift in this case is a strong function of the shock location making good shock resolution very important. The h mesh corresponds to figure 5 with every other mesh line removed. The corresponding solution is shown in figure 19. While both shocks are apparent in fig. 19 neither shock is very well defined due to the poor mesh resolution in the shock regions. Note the leading edge total pressure loss characteristic of the earlier results and poor resolution of the total pressure jump across the shocks. Figure 20 shows the corresponding embedded mesh used for this case where four embedded regions have been added to resolve the leading and trailing edges and the two shock regions. Comparing the embedded mesh solution shown in figure 21 with the h global solution shows much better resolution of the two shocks and a reduction in the surface total pressure losses which occur in the expansion region around the leading edge. Note in this example that the upper surface shock wave crosses the boundary of the embedded mesh region. Other than local loss of resolution (fig 21c) no difficulties are encountered. The convergence histories in terms of multiple-grid cycles between the h and embedded solutions are very similar (fig. 22).

The second test case to be shown is the RAE2822 supercritical airfoil (ref. 7) at $M_\infty = 0.75$ and $\alpha = 3.0$ deg.. At these conditions there is a very rapid expansion around the leading edge and also a strong shock at 80% chord on the upper surface. The embedded mesh used is presented in fig. 23 using embedded regions around the leading and trailing edges and in the shock region. The corresponding embedded mesh solution is shown in fig. 24. The higher grid resolution in the leading edge region using the embedded mesh is important to resolve the rapid expansion and nearly halves the total pressure errors in this region. The convergence rates for a global h mesh and the embedded mesh calculations are shown in fig. 25.

The final case to be presented is the Garabedian and Korn supercritical airfoil(ref. 18) with a design condition of $M_\infty = 0.75$ and $\alpha = 0.12$ deg. and a theoretical lift coefficient of 0.63. At design conditions the supersonic region extends over about 60% of the the upper surface. The solution in this case is very sensitive to the location of the sonic line in the flow. The embedded mesh used for this case is shown in fig. 26. The corresponding embedded mesh solution is shown in fig. 27 with a lift coefficient of 0.604. In cases where a known solution exists for comparison, it is found in this work that the calculated lifts are a few percent low. For this case while the embedded meshes reduced the total pressure error by almost half the effect on the lift coefficient was not very significant. Even with the embedded mesh regions the calculated lift is below the design conditions of ref. 18. The convergence histories are similar to others which have been shown.

CONCLUSIONS

The method for embedding meshes in a multiple-grid structure presented in this paper represents the first step in the development of a general modular approach to solving complex flow problems. While the present formulation uses a Lax-Wendroff type time marching scheme, the multiple-grid structure is a much more fundamental concept which need not be limited to this scheme. When this multiple-grid structure is viewed in terms of a pointer system the extension from a global grid structure to very general grid structures which include embedded mesh regions presents very little added complexity. The present pointer system was chosen for its flexibility, the cell being the most fundamental element of the grid struture. It is quite possible that for certain applications, such as vectorized algorithms, another pointer formulation might be more desirable. The addition of a pointer system does add to the total storage required per grid point of the system but for a proper distribution of mesh points, made possible by the general multiple-grid structure, the total number of grid points can now be minimized. This can then result in a significant reduction in storage over that required for an equivalent global grid. In addition, this reduction in total number of required mesh points then results in much less computational work, both due to the reduction in the number of points and also improved convergence rates. As shown by the presented cases, with the present formulation it has been possible to gain the fine mesh resolution within the embedded mesh regions while preserving the global coarse mesh convergence rates.

ACKNOWLEDGEMENTS

This work was partially supported by the Air Force Office of Scientific Research under grant AFOSR-82-0136, Dr. J. Wilson technical monitor, and by the National Aeronautics and Space Administration under grant NGT 22-009-901, Dr. Randolph Graves, technical monitor.

REFERENCES

1. Ni, R.H., "A Multiple-Grid Scheme For Solving the Euler Equation," AIAA Journal, Vol. 20, No. 11, November 1982, pp 1565-1571.
2. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions to the Euler Equation by Finite Volume Methods Using Runge-Kutta Time Stepping," AIAA Paper 81-1259, June 1981.
3. Rizzi, A., "Damped Euler Equation Algorithms to compute Transonic Flow Around Wing-Body Configurations," AIAA Journal, Vol. 20, No. 10, October 1982, pp 1321-1328.
4. Thompson, J.F., "A Survey of Grid Generation Techniques in Computational Fluid Dynamics," AIAA Paper 83-0447, January 1983.
5. Boppe, C.W., "Computational Transonic Flow about Realistic Aircraft Configurations," NASA CR 3243, May 1980.
6. Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," Mathematics of Computation, Vol. 31, No. 138, April 1977, pp 333-390.
7. Brown, J.J., "A Multigrid Mesh-Embedding Technique for Three-Dimensional Transonic Potential Flow Analysis," NASA CP 2202, October 1981.
8. Usab Jr., W.J., "Embedded Mesh Solutions of the Euler Equations Using a Multiple-Grid Method," M.I.T. Doctoral Thesis (in preparation).

9. Johnson, G.M., "Convergence Acceleration of Viscous Flow Computation," NASA TM 83039, October 1982.
10. Abbett, M.J., "Boundary Condition Calculation Procedures for Inviscid Supersonic Flow Fields," AIAA Computational Fluid Dynamics Conference Proceedings, Palm Springs, California, July 19-20, 1973.
11. McCartin, B., "Theory, Computation, and Application of Exponential Splines," Courant Mathematics and Computing Laboratory, U.S. Dept. of Energy Report No. DOE/ER/03077-171, October 1981.
12. Whitfield, D.L., Thomas, J.L., Jameson, A., and Schmidt, W., "Computation of Transonic Viscous-Inviscid Interacting Flow," Proceedings of Second Symposium of Numerical and Physical Aspects of Aerodynamic Flows, California State University, January 1983.
13. Ludford, G.S.S., "The Behavior at Infinity of the Potential Function of a Two Dimensional Subsonic Compressible Flow," Journal Math. Physics, Vol. 30, 1951, pp 117-130.
14. Eriksson, L.E., "Generation of Boundary-Conforming Grids Around Wing-Body Configurations Using Transfinite Interpolation," AIAA Journal, Vol. 20, No. 10, October 1982, pp 1313-1320.
15. Lock, R.C., "Test Cases for Numerical Methods in Two-Dimensional Transonic Flows," AGARD Report No. 575, November 1970.
16. Thompkins Jr., W.T., Tong, S.S., Bush, R.H., Usab Jr., W.J., and Norton, R.J.G., "Solution Procedures for Accurate Numerical Simulations of Flow in Turbomachinery Cascades," AIAA Paper No. 83-0257, January 1983.
17. Rizzi, A., "Numerical Methods for the Computation of Inviscid Transonic Flows with Shock Waves," A GAMM Workshop, Vieweg und Sohn, Wiesbaden, 1981.
18. Kacprzynski, J.J. and Ohman, L.H., "Analysis of the Flow Past a Shockless Lifting Airfoil in Design and Off-Design Conditions," National Research Council of Canada, Aeronautical Report LR-554, November 1971.

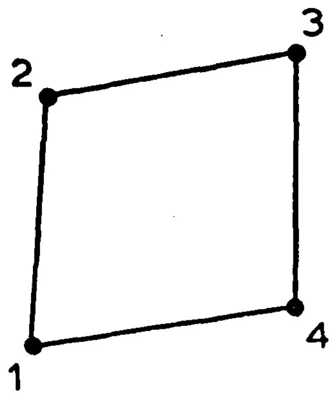


Figure 1. Base solver cell notation .

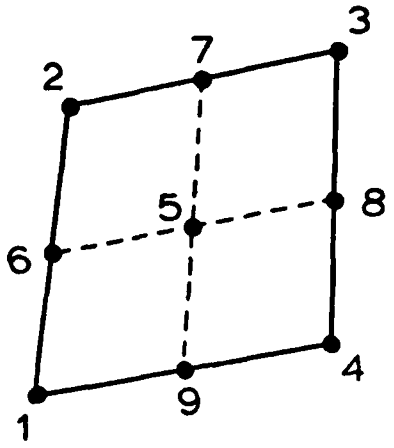


Figure 2. Accelerator cell notation .

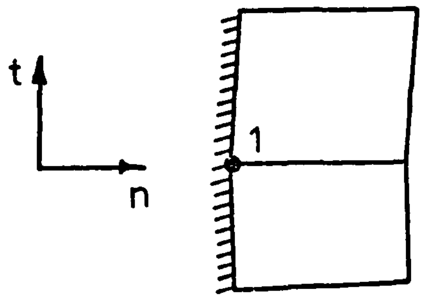


Figure 3. Boundary cell notation.

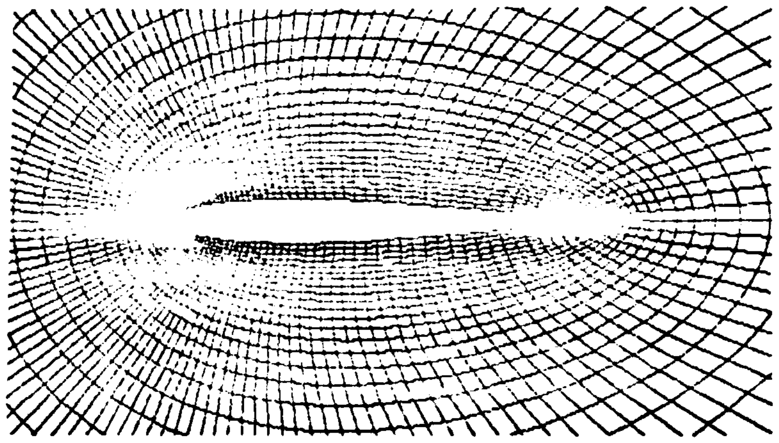


Figure 4. NACA0012 airfoil 129*33 global mesh.

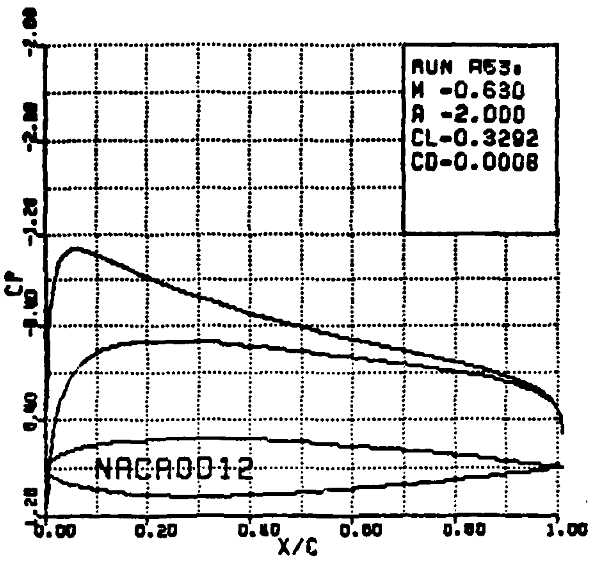


Figure 5a. Surface pressure coefficient.

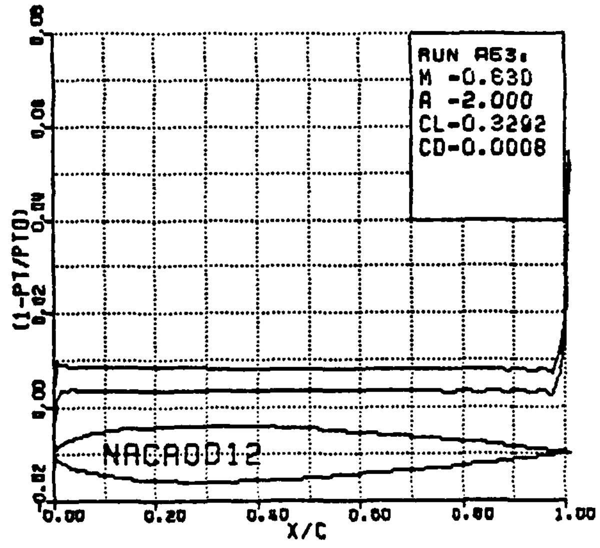


Figure 5b. Surface total pressure loss.

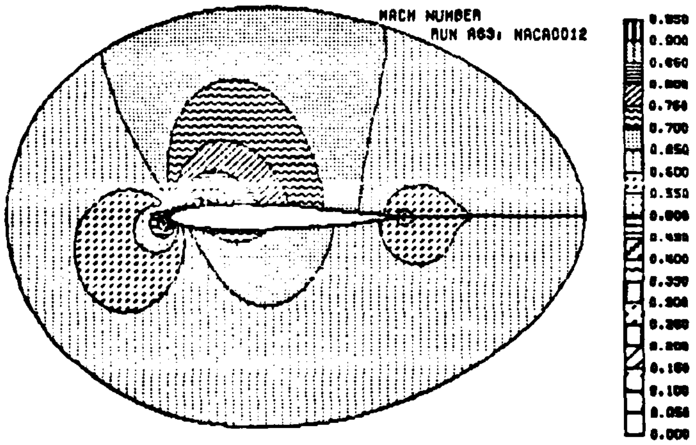


Figure 5c. Mach number contours.

Figure 5. NACA0012 airfoil for $M_\infty = 0.63$ and $\alpha = 2.0$ deg. on a 129*33 mesh.

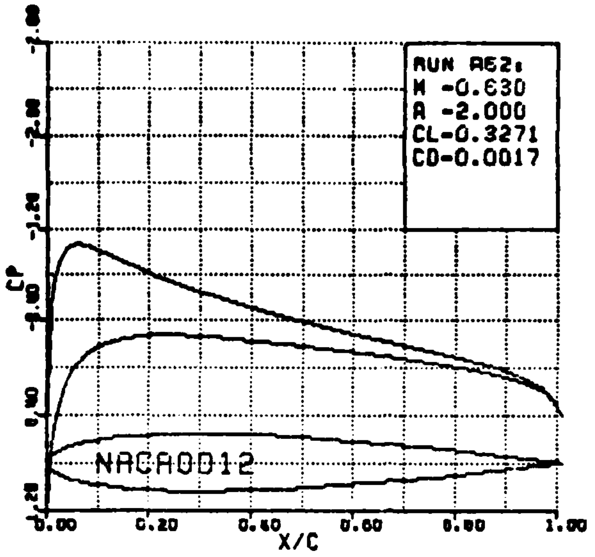


Figure 6a. Surface pressure coefficient.

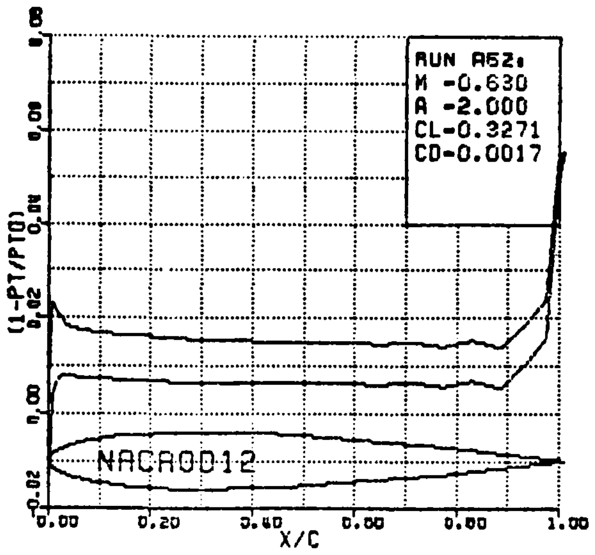


Figure 6b. Surface total pressure loss.

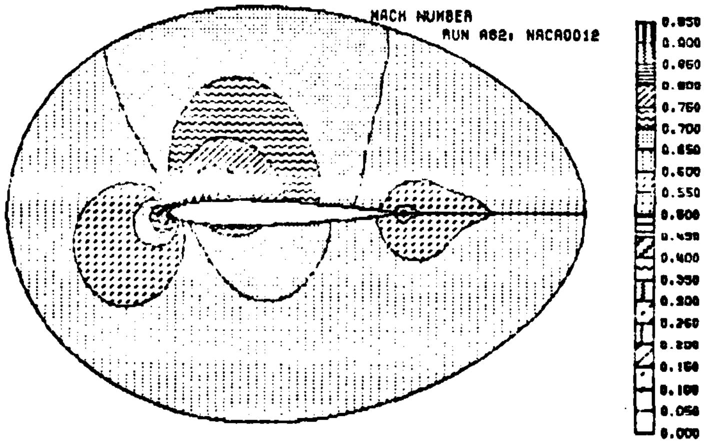


Figure 6c. Mach number contours.

Figure 6. NACA0012 airfoil for $M_\infty = 0.63$ and $\alpha = 2.0$ deg. on a 65*17 mesh.

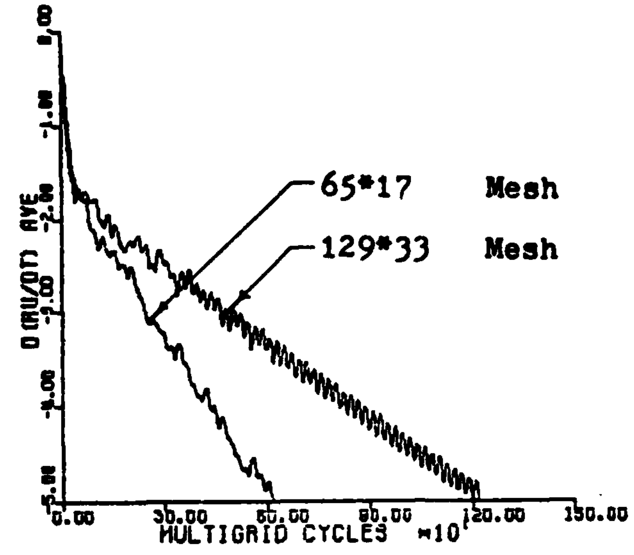


Figure 7. Convergence histories for NACA0012, M = 0.63 and = 2.0 deg.

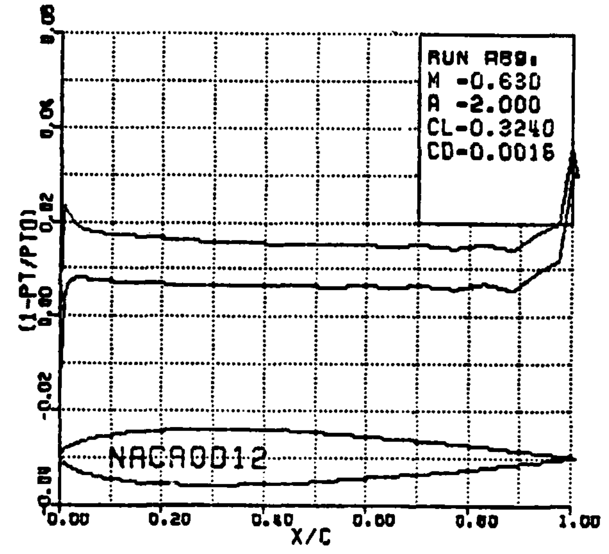


Figure 8. Surface total pressure loss for 65*17 mesh with reduced skewness near trailing edge.

Theoretical $C_L = 0.335$

Far Field B.C.	C_L R=5	C_L R=10
Uniform Flow	0.2685	0.2973
Vortex Flow	0.3270	0.3274

Table 1: Effect of far field boundary condition and location on calculated lift.

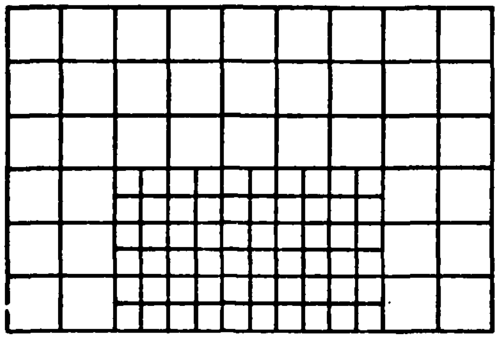


Figure 9. Embedded mesh topology.

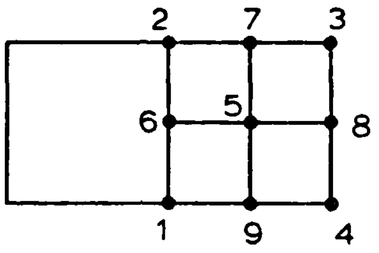


Figure 10. Embedded mesh interface notation.

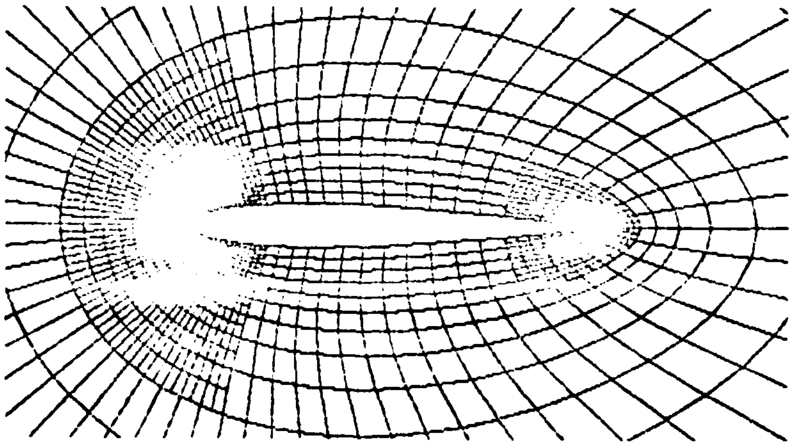


Figure 11. Leading and trailing edge embedded mesh regions for a NACA0012 airfoil.

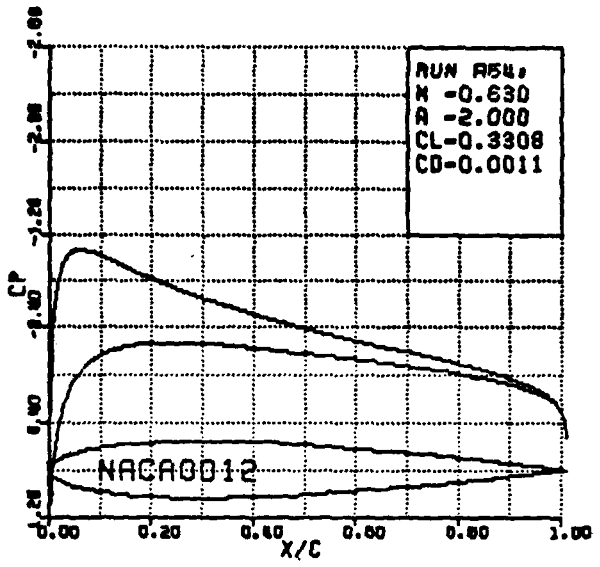


Figure 12a. Surface pressure coefficient.

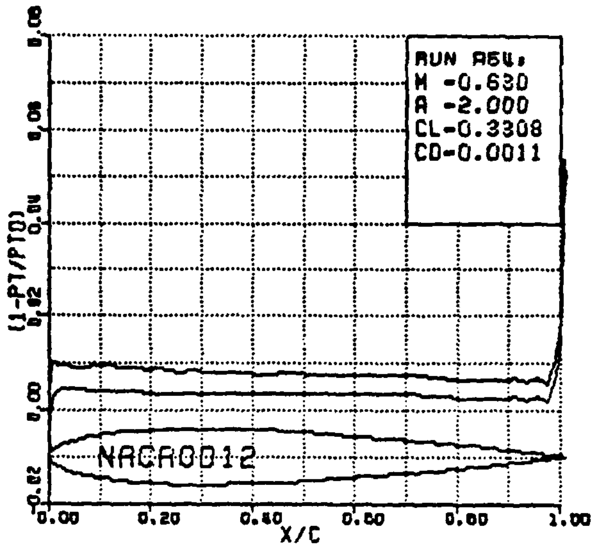


Figure 12b. Surface total pressure loss.

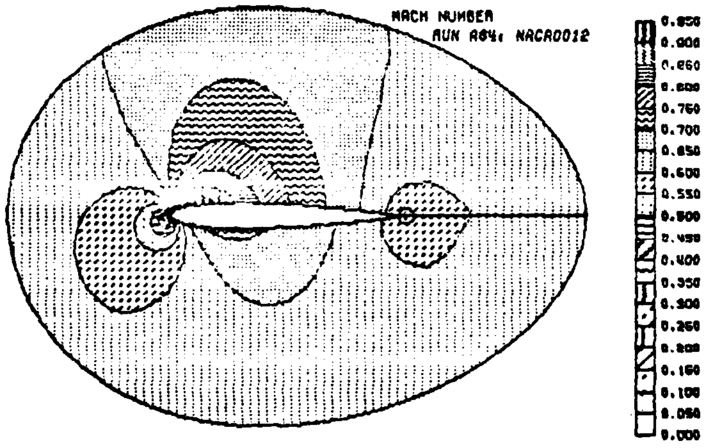


Figure 12c. Mach number contours.

Figure 12. NACA0012 airfoil for $M_\infty = 0.63$ and $\alpha = 2.0$ deg. on embedded mesh of figure 11.

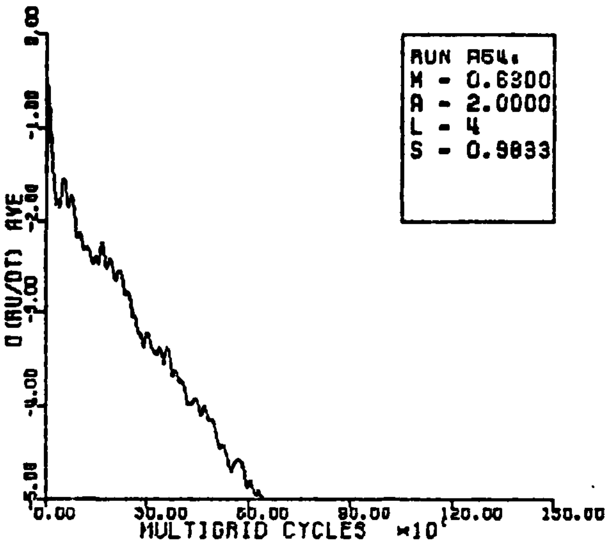


Figure 13. Embedded mesh convergence history.

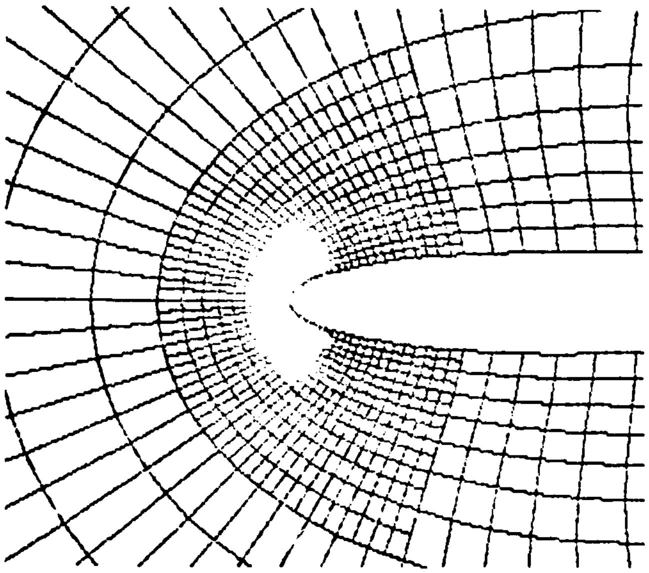


Figure 14. Double embedded mesh in the leading edge region.

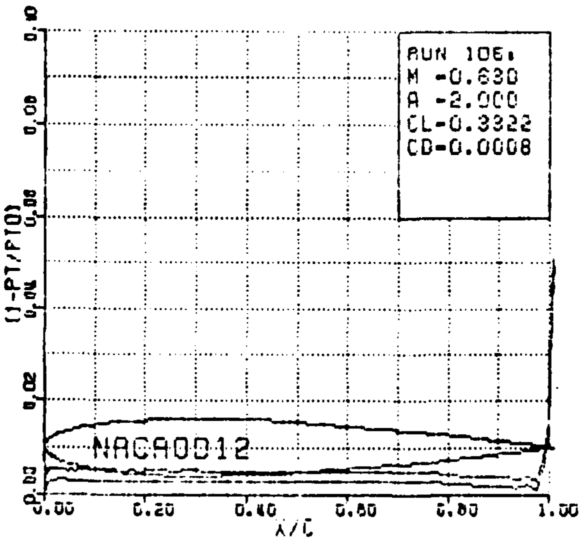
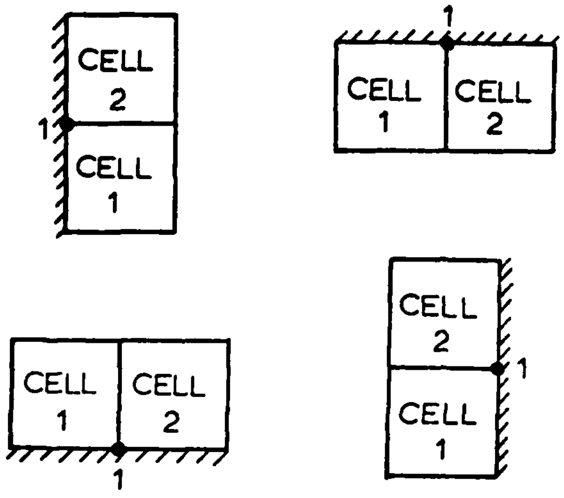


Figure 15. Surface total pressure loss for double embedded leading edge mesh.



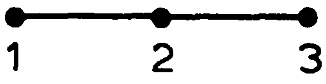


Figure 17. Embedded mesh interface pointer notation.

Figure 16. Four possible boundary cell orientations.

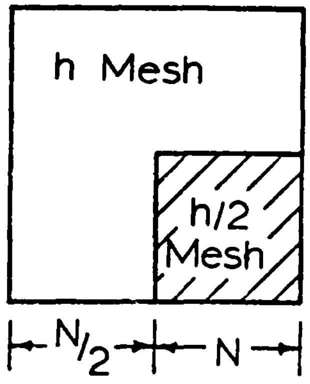


Figure 18a.

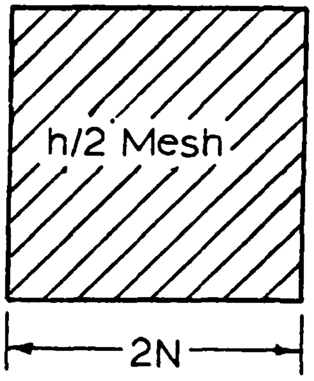


Figure 18b.

Figure 18. Embedded and global mesh refinememt example.

Table 2: Storage requirements for example in figure 18.

	Global h/2 Mesh	Embedded h/2 Mesh
No. Mesh Points	$4N^2$	$1.75N^2$
Solution Storage	$80N^2$ Words	$35N^2$ Words
Pointer Storage	0 Words	$18(N-1)^2$ Words
Total Storage	$80N^2$ Words	$53N^2-36N+1$ Words

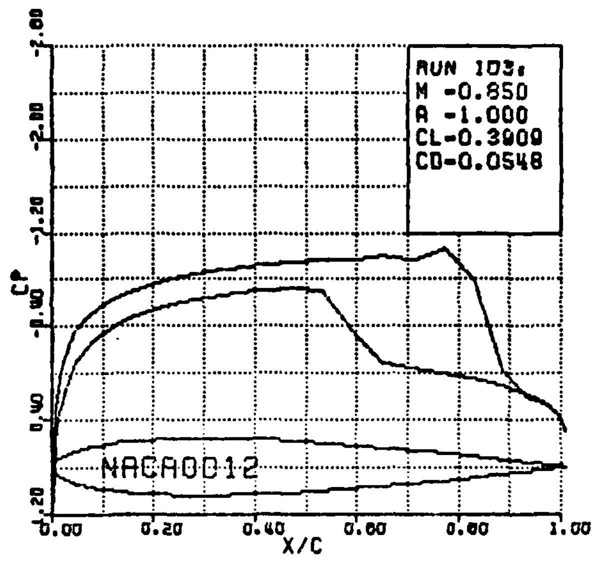


Figure 19a. Surface pressure coefficient.

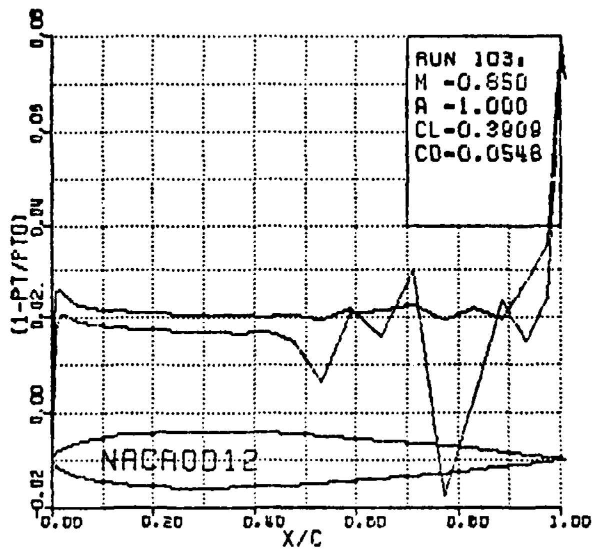


Figure 19b. Surface total pressure loss.

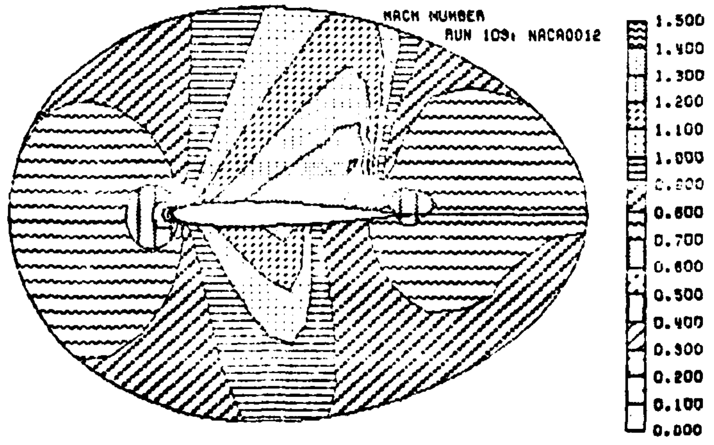


Figure 19c. Mach number contours.

Figure 19. NACA0012 airfoil for $M_\infty = 0.85$ and $\alpha = 1.0$ deg. on 65*17 mesh.

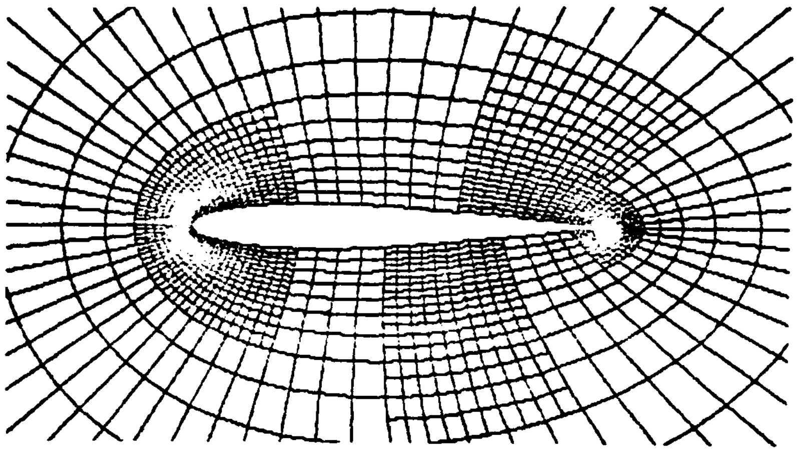


Figure 20. Embedded regions, NACA0012 airfoil for $M_\infty = 0.85$ and $\alpha = 1.0$ deg.

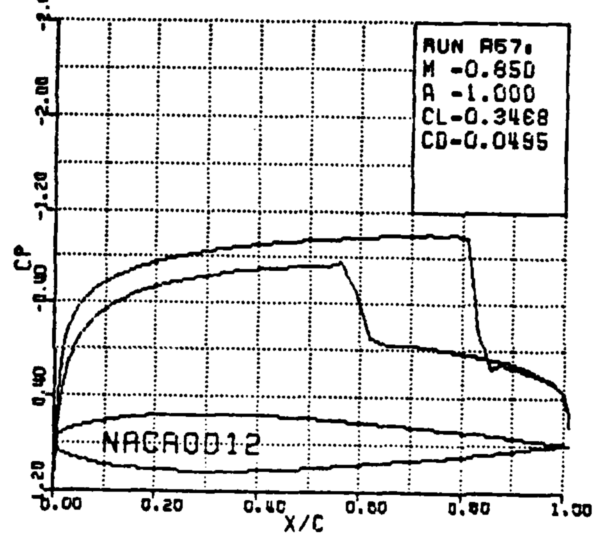


Figure 21a. Surface pressure coefficient.

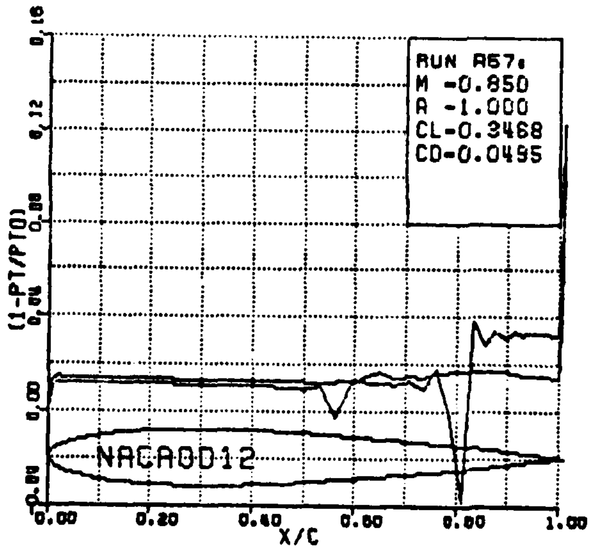


Figure 21b. Surface total pressure loss.

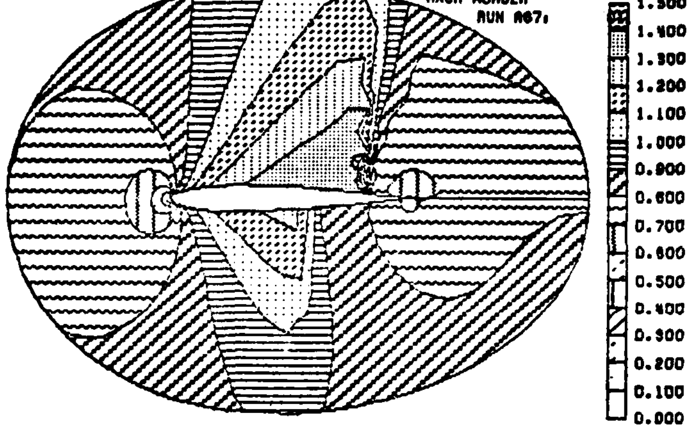


Figure 21c. Mach number contours.

Figure 21. NACA0012 airfoil for $M_\infty = 0.85$ and $\alpha = 1.0$ deg. on embedded mesh of figure 20.

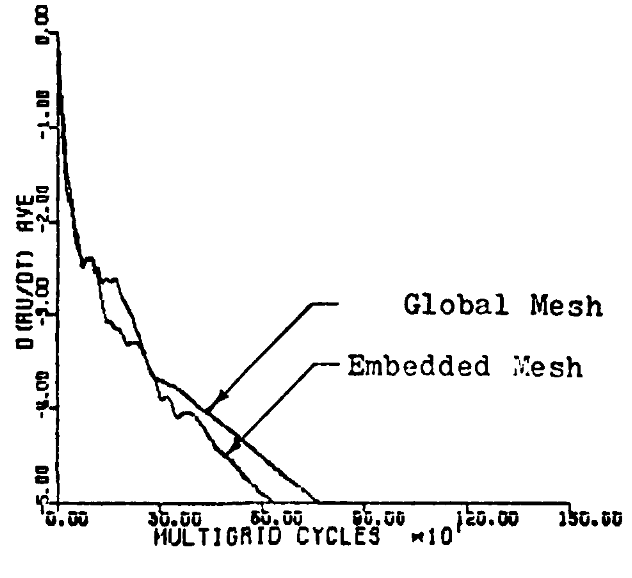


Figure 22. Convergence histories for figures 19 and 21.

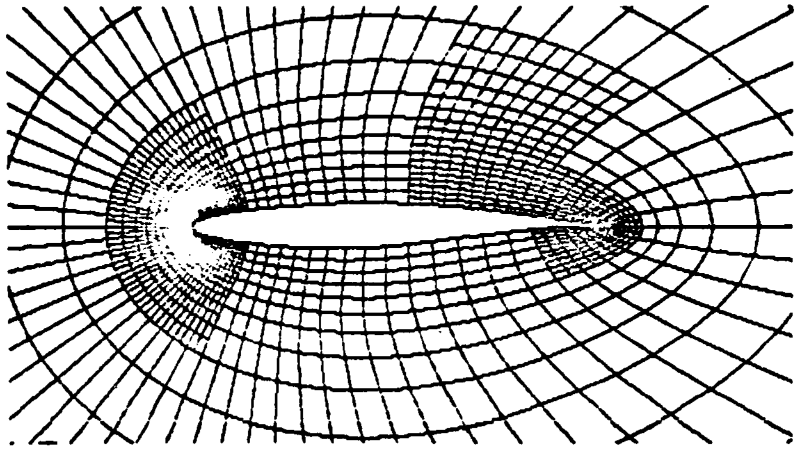


Figure 23. Embedded regions, RAE2822 airfoil for $M_\infty = 0.75$ and $\alpha = 3.0$ deg.

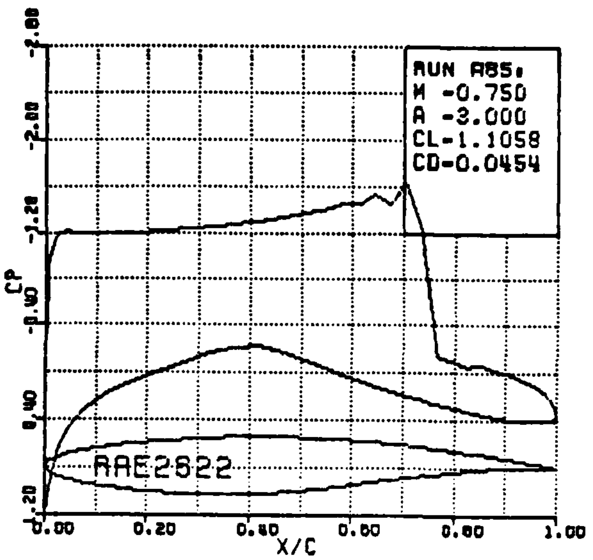


Figure 24a. Surface pressure coefficient.

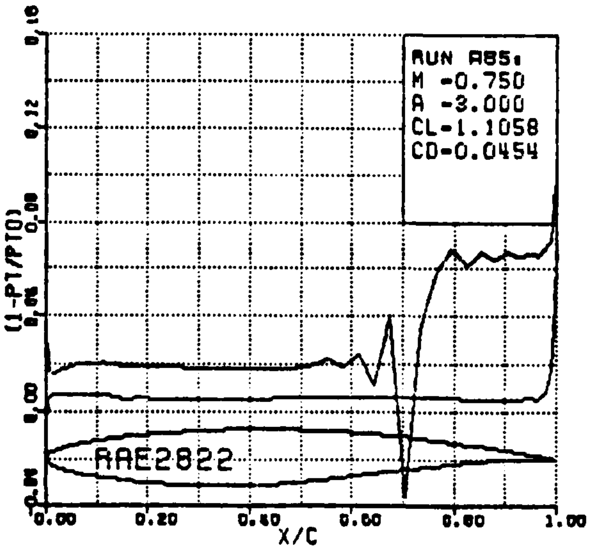


Figure 24b. Surface total pressure loss.

Figure 24. RAE2822 airfoil for $M_\infty = 0.95$ and $\alpha = 3.0$ deg. on embedded mesh of figure 23.

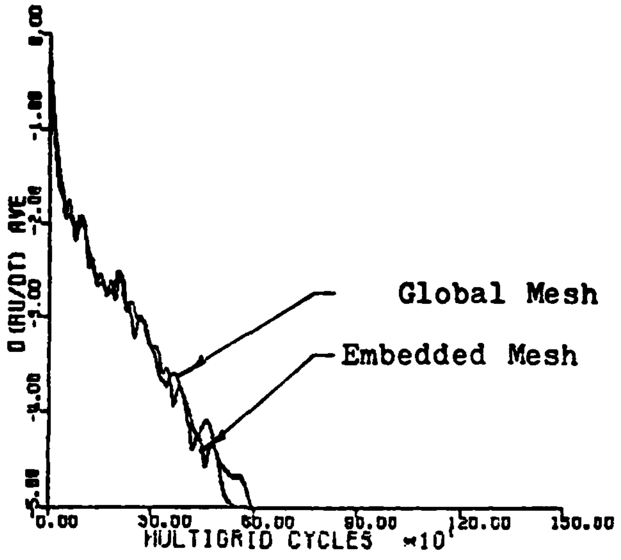


Figure 25. Convergence histories for global mesh and embedded mesh of figure 24.

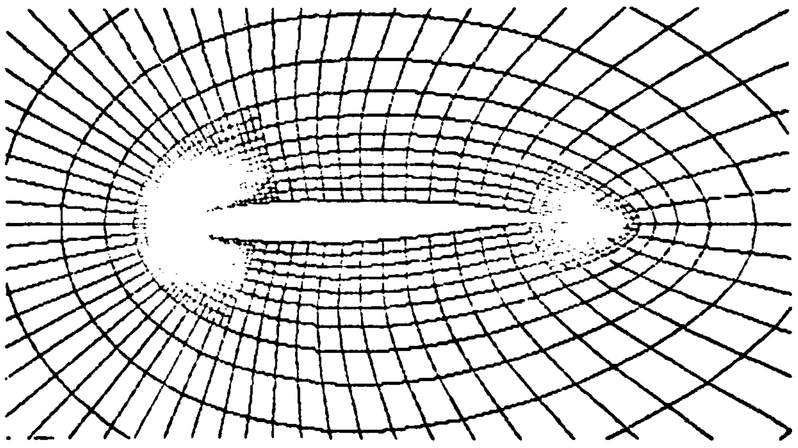


Figure 26. Embedded mesh regions, Korn airfoil for $M_\infty = 0.75$ and $\alpha = 0.12$ deg.

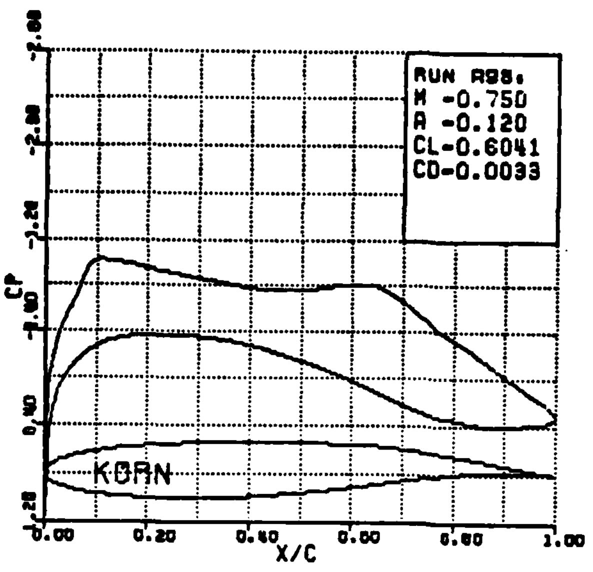


Figure 27a. Surface pressure coefficient.

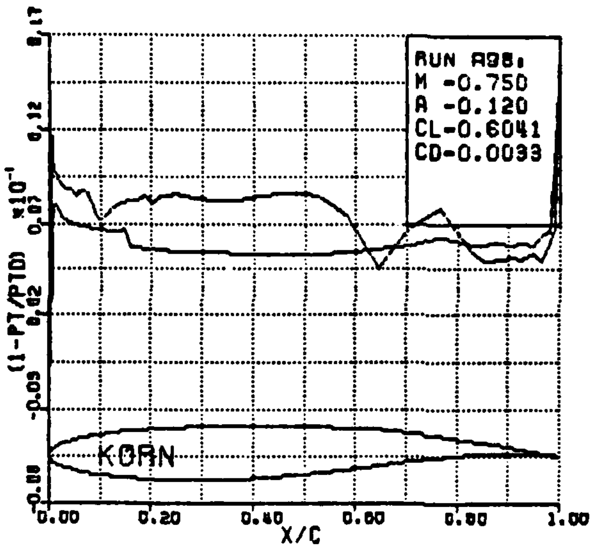


Figure 27b. Surface total pressure loss.

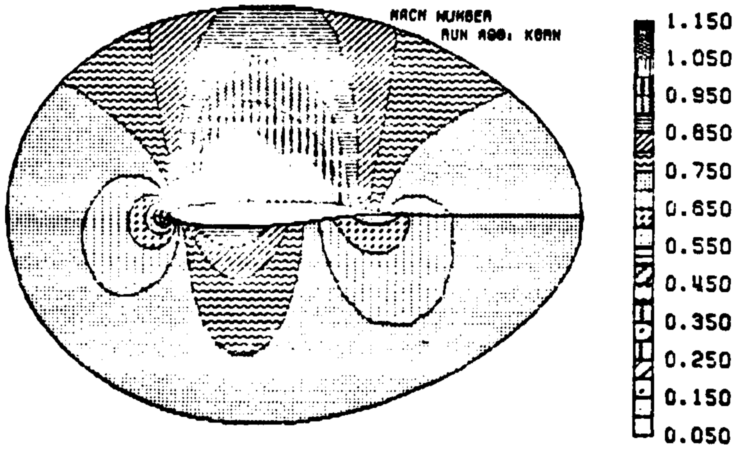


Figure 27c. Mach number contours.

Figure 27. Korn airfoil for $M_\infty = 0.75$ and $\alpha = 0.12$ deg. on embedded mesh of figure 26.

APPENDIX B

ADAPTIVE EMBEDDED SUBDOMAINS

1. Background

The objective of this effort is to develop adaptive solution algorithms for the Navier-Stokes equations. A typical and intended application is the computation of the viscous, compressible flow over an isolated lifting airfoil at a realistic flight Reynolds number.

Adaptive solution algorithms is a generic designation for numerical methods which sense some unique physical behavior in the flowfield during computation. The procedure subsequently changes the governing equations and/or computational grid so as to adequately describe the primary features. Adaptive solution algorithms which change the governing equations are referred to as adaptive equation techniques; those which alter the computational mesh are known as adaptive grid techniques.

In order to understand the usefulness of these kinds of algorithms, it is helpful to describe the concept of features. A feature is defined as a region of the flowfield which delineates dominant physics that is significantly different from the physics of the surrounding flow. Relevant examples of features include shock regions, boundary layers, wakes, and embedded vortices. Each feature can be associated with a number of characteristics--i.e. type, location, strength, orientation, and scale. The scale of a feature has special importance in classifying the feature to be either collapsible or non-collapsible. The collapsible character of a feature implies that the scale is sufficiently small relative to the global scale such that the entire feature may be collapsed to a point or line. Necessarily, it is implied that the physical description associated with the events within the feature are properly taken into account despite the collapse modeling. Examples of collapsible features are shocks and thin boundary layers.

If a feature is collapsible, adaptive equation techniques are very useful. In adaptive equation schemes, a special subset of the governing equations is used to adequately model the physics within the feature. This may take the form of a jump condition. An inviscid/viscous interaction algorithm is a familiar technique which alternates between system descriptions. In the viscous example the influence of viscosity is treated as if its effects are virtually confined to an infinitesimally thin region adjacent to a wall (or possibly center line of a wake). An outer flow is computed as if it were inviscid, and extended to the wall; an inner viscous (boundary/shear layer) solution is computed simultaneously and the two solutions are suitably matched. The essential point is that different subsets of the Navier-Stokes equations are relevant and used in each of the two regions, consistent with the local physics. Shock fitting is another example of an adaptive equation algorithm and one which intuitively fits the collapsible concept. Shock jump relations are used at such a feature and those special equations allow for a discontinuity in the computed flowfield and a proper matching of the flow variables on both sides of the jump. In each case, viscous or discontinuity, the algorithms rely on a simplifying change of the governing equations at the feature to take into account the unique physics. However, the proper matching conditions are not at all obvious in most cases, and this is a major obstacle.

Unfortunately, some features are not collapsible, and adaptive equation algorithms cannot always be used. Instead, adaptive grid algorithms may use the same descriptive equations throughout the flowfield but with the grid spacing varied so as to properly capture the local physics. A viscous flow example would be an adaptive grid algorithm which captures the boundary layer adjacent to a wall by decreasing the local mesh spacing in order to resolve the local viscous behavior. Within the basic framework of adaptive grid algorithms, there are currently two major approaches. The first (and currently most popular) involves grid point redistribution; the second can be called embedded grids.

In grid point redistribution, a fixed number of grid points are spaced throughout the flowfield. Grid point movement is induced by regions of larger error (most likely regions near futures) which attract mesh lines. Advocates describe the resulting solution as being uniformly good. Alternatively it may be viewed to be uniformly bad, since in those regions with small initial errors the removal of grid points results in larger errors. The method is very popular, however, since the logic needed to implement the algorithm is relatively straightforward. The same logic can be used for all cases, independent of the features which are present (or absent).

The other adaptive grid algorithm (embedding) maintains the global grid as an invariant and furnishes additional grid points at the features. This yields locally embedded patches that increase the accuracy at features, and simultaneously maintains the global mesh accuracy. Note that this does involve increasing the total number of computational nodes and therefore both computation effort and required storage. With careful control of the adaptation algorithm, however, it does prove to be possible to keep this under control. The chief disadvantage is that fine regions exist only as a result of local embedding and thus introduces artificial internal boundaries. The necessary coupling of the global and embedded grids can cause problems. The logic needed to implement an embedding method is considerably more complicated than that for grid point redistribution.

2. General Approach

The approach taken in this research task is to allow both adaptive equation and adaptive grid algorithms. The combination offers a maximum degree of computational efficiency with appreciable generality.

As an example, consider the compressible, viscous flow over an isolated airfoil at a high angle of attack and a typical flight Reynolds number. The expected features are shown in Figure 1. The shock and the boundary layer on the lower surface are collapsible, and thus amenable to adaptive equation techniques. On the upper surface, however, the shock induced separation results in a viscous region with a scale which is on the order of the airfoil thickness and clearly non-collapsible. Thus in that region an adaptive grid scheme is required.

It is helpful to describe the overall adaptive solution algorithm in terms of its sequential steps:

1. Compute a solution on the given global grid.

2. On the second or later computation cycle examine specified key parameters (e.g. lift coefficient, or wave drag) for the extent of change since the previous cycle; STOP when criteria is met.

3. Search entire flowfield for possible feature points.

4. Cluster individual feature points into feature groupings.

5. For each feature:

   - If not collapsible, adapt the grid and proceed to 6;
   - If collapsible, search an included library of characteristics until
     - either a match is found (enabling use of the prescribed adapted equations and contraction of the adapted grid)
     - or a match is not found (and the grid is adapted further).

6. Cycle through 5 to consider all features.

7. Return to 1.

Each of the above steps will now be described in more detail.

Solve equations system. The basic field evaluations used in this study have made use of a conservative, finite-volume, time-marching scheme developed by Ni and subsequently modified by Usab in the present parallel task to solve the two-dimensional unsteady governing equation in the vector form

$$\frac{dU}{dt} + \frac{dF}{dx} + \frac{dG}{dy} = \frac{dH}{dx} + \frac{dI}{dy} \tag{1}$$

where F and G are the convective, and H and I are the diffusive terms. This scheme is essentially a Lax-Wendroff one-step procedure with a multiple grid accelerator. Since it is explicit, the computation for each control volume is independent of other control volumes. Thus the use of adaptive equations is compatible with imposing alternate physics within different cells, and the computation algorithm can be implemented easily. In addition, the multiple grid accelerator procedure can be used to couple the global and embedded meshes as described previously for Task I.

Search for feature points. An adaptive solution technique requires some measure for the existence of a feature within the domain. Typically, this corresponds to evidence of large change in the value of an important "variable." The "change" can be deduced from such criteria as a gradient, second difference, local truncation error, or other parameters which are indicators of nonuniform fields. "Variable" refers to any flow state property (e.g. density or velocity), or an integral parameter (e.g. displacement thickness), or a derived quantity (e.g. vorticity).

Clustering. Feature points are grouped so that those with the same dominant physics can be examined as a whole. For example, for the flow shown in Figure 1, those feature points associated solely with the shock need be considered simultaneously.

Grid Adaptation. The simplest implementation of a grid adaptation algorithm is the subdivision of each cell which contains a node that has been designated as a feature point. This is the algorithm that is currently being used, but it has been found to be slightly inadequate since it allows for multiply connected adapted regions with "holes." Somewhat more complicated algorithms do exist for handling such a possibility and are being investigated at the current time. It is also possible to contract cells (delete all fine cells associated with the chosen cell) and this is an especially important capability when used in conjunction with adaptive equations. A pointer system similar to that described under Task I has been used for all procedures in the adaptive work. However, it was necessary to modify the system somewhat in view of the dynamic addition and subtraction of grid cells. In effect, the adaptation freedom of cell placement insists on a more complex pointer system which includes additional information.

Characteristics library. The type of a collapsible feature is quite necessary and must be established since appropriate equations must be assigned. This can be viewed as a classical pattern recognition problem and may be solved by comparing the characteristics of the feature with those stored in a library. For example, the library might contain the following description (constraints) of a shock: very thin, with a jump in normal velocity component given by the Rankine-Hugoniot conditions and the components being supersonic upstream and subsonic downstream. Similar constraints for boundary layers, wakes, embedded vortices, and stagnation points must be provided in a library for those and any other candidate physical events which are subject to adapted equation assignment.

Equation Adaptation. It is assumed that appropriate algorithms do exist for all kinds of flow physics that are of interest (e.g., inviscid flow, shock fitting, boundary layer). Adapting the equations reduces to a bookkeeping problem, but the essential requirement is that each cell be flagged to indicate the type of computation that is appropriate. Each type of computation also must be constructed in such a way that it can be easily adjoined to the basic scheme in a most general way.

3. Statement of Work

The objective of this task has been to construct a scheme which is able to find the extant flow features and then to apply the most efficient algorithm to each of them independently. In order to proceed in an orderly fashion the overall development has been subdivided into the following subtasks:

- 1-D Euler flow with adaptive grid
- 2-D model problem with adaptive grid
- 1-D Euler flow with adaptive equation (shock fitting)
- 2-D model problem with adaptive equation (thin layer)
- 2-D Navier-Stokes with adaptive grid
- 2-D Navier-Stokes with adaptive solution

The first has been completed, and the second is nearly complete. Some results for each of these subtasks are described in more detail below.

4. 1-D Euler Flow with Adaptive Grid

The initial major objective was to determine the level of difficulty associated with finding simple features and subsequently adapting the grid. The same code developed for this subtask should also prove to be useful in studying the interaction of simultaneous adaptive grid and adaptive equation algorithms (e.g. 1-D Euler with shock fitting).

The quasi-one-dimensional Euler equations are given in vector form by

$$\frac{dU}{dt} + \frac{dF}{dx} = G \tag{2}$$

where

$$U = \begin{vmatrix} \rho \\ \rho u \\ \rho h_0 - p \end{vmatrix} , \quad F = \begin{vmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_0 \end{vmatrix} , \quad G = \begin{vmatrix} \rho u \\ \rho u^2 \\ \rho u h_0 \end{vmatrix} \frac{\partial \ell n A}{\partial X} \tag{3}$$

and t and x denote time and axial position respectively. The equations have been solved using Ni's conservative finite volume, multiple-level, time-marching scheme. Characteristic boundary conditions are applied at both the inlet and exit and embedded regions are computed using a variant of Usab's embedded mesh procedure.

Figure 2 shows the solution for a divergent duct with a supersonic inlet and sufficient back pressure to induce a standing shock. The computed solution is plotted as a line and the analytic shock position and jump levels are denoted by the symbols. The duct and grid are shown below. The Mach number distribution computed on the global grid (level 0) agrees quite well with the analytic solution except in the vicinity of the shock, where the computed shock thickness extends over roughly two computational cells and pre- and post-shock oscillations are evident. The adaptive grid algorithm was then employed such that all computational cells found to fall within the feature were subdivided. For this example, feature points were defined to be those points for which the second difference of the density exceeded the average second difference. This subdivision process resulted in the grid geometry shown at the bottom of Figure 3. For simplicity, the new cells, which are half as large as the original cells, are termed level 1 cells. The corresponding Mach number distribution is shown at the top of Figure 3, and it is apparent that the shock width has been approximately halved and the oscillations associated with the shock have been confined to a smaller portion of the domain. The same process when applied again yields the results shown in Figure 4. The finest cells in this case are referred to as level 2 cells. Again the shock width and region of oscillations have been halved.

As a further check on the differences between solutions that may have resulted from a finer resolution only near the shock or other unknown embedded mesh influences, the same field was computed with level 2 cells everywhere throughout the duct. The very good agreement with Figure 4 (i.e. the two solutions virtually coincide) strongly suggests that the improved resolution at the feature is indeed responsible for the improvement and that the internal boundaries do not introduce extraneous effects.

Figures 5, 6 and 7 provide a second example of computed solutions representing a Laval nozzle with a subsonic inlet, expansion through sonic speed at the throat, and containing a shock in the diverging portion as a result of an implied back pressure. Cases with a uniform mesh, one level of adaptation (level 1), and two levels of adaptation (level 2) are shown as indicated by the accompanying channel and grid. Again the adapted solutions are clearly superior.

Since the purpose of adaptive solution algorithms is to efficiently provide accuracy, it is proper to compare computation times for the adapted and non-adapted runs. The computation times for adapted runs should in fairness include all of the time for the initial solution, feature finding, grid adaptation, and the iterations to final convergence. Table I compares normalized CPU times for each of the above geometries. The first row entries are for the initial global (level 0) geometry. The second row contains times for the same solution accuracy near the shock but completed in global (every cell a level 1 cell) and separately in adapted (only embed level 1 near the shock) fashion. A similar comparison is made on row 3 for level 2 cells. For both geometries, the adapted grid solution requires less computer time. The Laval nozzle is seen to be even more efficient than the divergent duct case, which may be attributed to the fact that the embedded region is in that case a much smaller part of the global grid. It is tempting to extrapolate this to an airfoil situation in which the shock is expected to cover an even smaller part of the flowfield; on that basis it appears reasonable to anticipate a tenfold or more saving.

Table I. Comparison of Computation Times

	Duct	Nozzle
Base (uniform level 0) grid	1.0	1.0
Level 1 grid Global/Adapted	2.3/1.8	3.1/1.8
Level 2 grid Global/Adapted	9.7/4.5	10.8/3.1

5. 2-D Model Problem with Adaptive Grid

Extension of the self-adapting procedures to two dimensions was thought to warrant an initial study based on a model system. Specifically, the clustering of feature points is much more difficult in two dimensions than in one dimension. This is compounded by the fact that two dimensional problems have features which may curve or even split (as in the case of a lambda shock). The primary purpose was an examination of feature topologies, and their manipulation and behavior during computational cycling. For that purpose the full Euler or Navier-Stokes equations were not considered necessary at this time. Instead, a two-dimensional model equation was used so as to allow for a more controlled environment and much more rapid solution times.

Multiple grid acceleration was employed for this two-dimensional model as in the one-dimensional duct flow cases. However, the 2-D problem in some instances resulted in multiple grid applications that actually decelerated the convergence rate. A detailed examination of the behavior is being made and an explanation appears plausible and inherent in the multiple grid algorithm.

The governing equation for the two-dimensional model problem was assumed to be

$$U_t + AU_x + BU_y = C\left(U_{xx} + U_{yy}\right) \tag{4}$$

where the subscripts denote differentiation and where A, B, and C at most depend on x and y. This scalar equation has both convective and diffusive terms and is quite suitable as a model equation for this subtask. By properly selecting combinations of the coefficients A, B and C, features can be made to have any desired orientation or curvature in the field. Solutions were obtained by the Ni scheme described above and with characteristic boundary conditions imposed.

Figure 8 shows a typical global grid (level 0) and field contours of the variable U for a case in which the coefficients A and B are chosen to be equal (implying convective propagation to proceed up and to the right, inclined 45° across the domain. The diffusive coefficient C was defined such that the disturbance at the left-hand boundary decreased in amplitude as it was convected. The initial distribution on the left edge was Gaussian. The third (lower) part of the figure indicates feature points by a "+" and others by a ".". Feature points were defined in this case as points at which the gradient of U is greater than the average gradient value in the field.

Figure 9 shows the resulting adapted grid and the corresponding contours of U. The corresponding level 1 global solution is shown in Figure 10. The outermost U contour from Figure 9 has been added to Figure 10 for comparison. The remaining inner four contours essentially coincide in both figures. The comparison discloses small errors which are introduced by the edge of the embedded region. The apparent waviness is associated with solving equations with diffusion contributions, which introduces new physics at the edges of the embedded regions. As a consequence there is a need for a slightly revised algorithm than is sufficient when only convection is present. Diffusive physics demands a new coupling of computational nodes across those edges for comparable accuracy in the modeling. An adequate algorithm for that purpose is part of the ongoing effort; although more work is required on this point, the errors remaining here are believed to be removable. Again, a significant CPU saving was realized by localizing the adapted level 1 region to the region of appreciable nonuniformity.

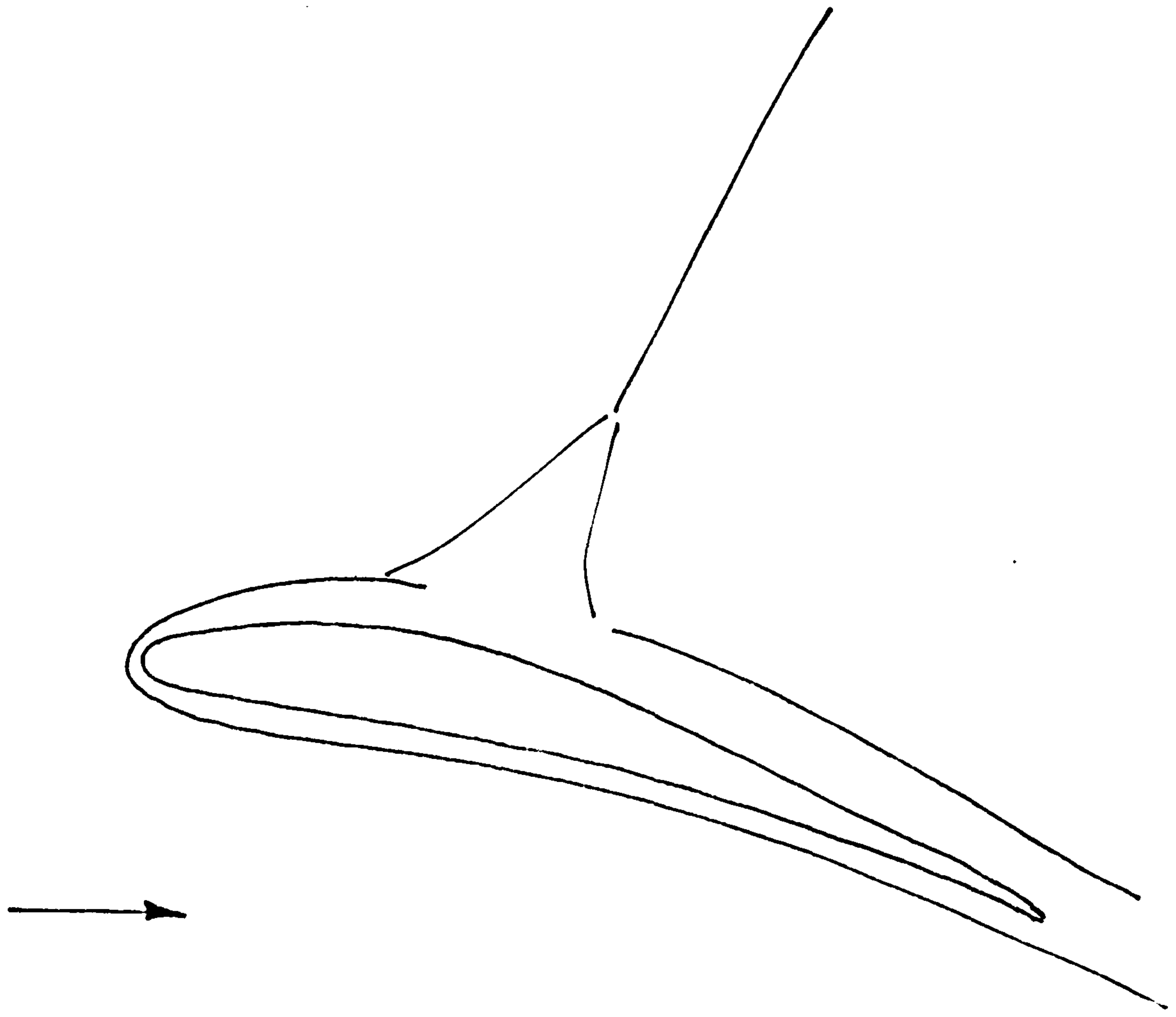
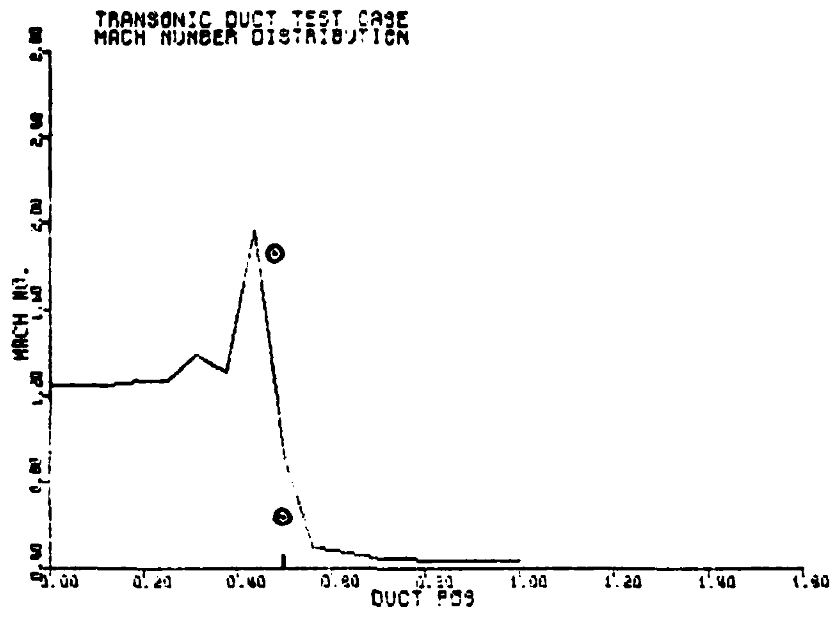


Figure 1 Typical airfoil flowfield features of boundary layer, trailing wake, shock and shock/viscous layer interaction



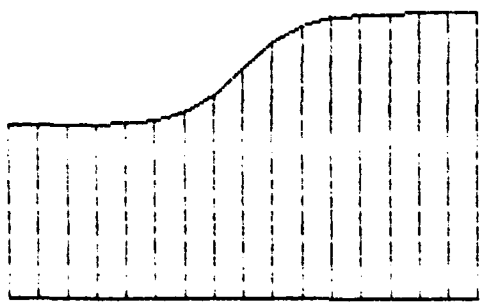
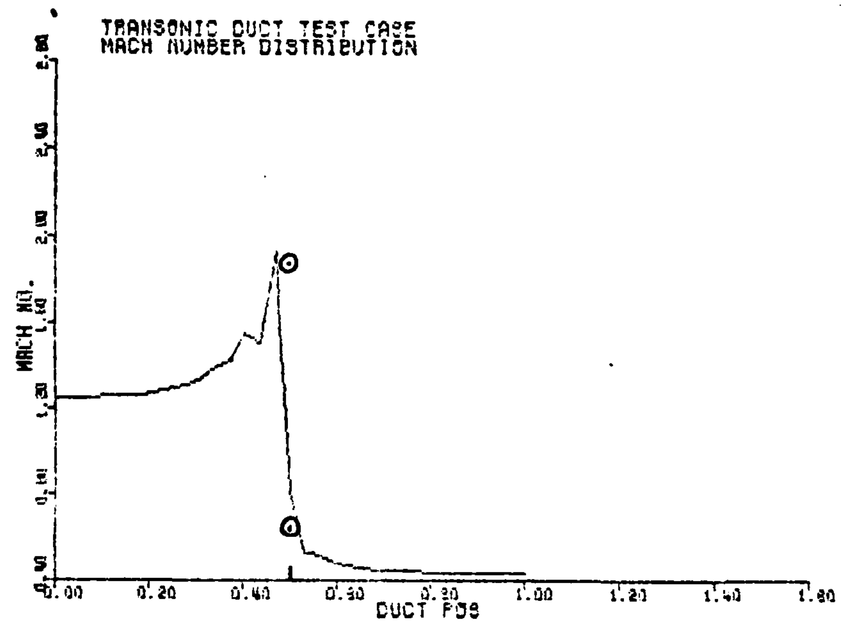


Figure 2 Mach number distribution for divergent duct flow with supersonic inlet/subsonic exhaust conditions

Level 0, Uniform global grid



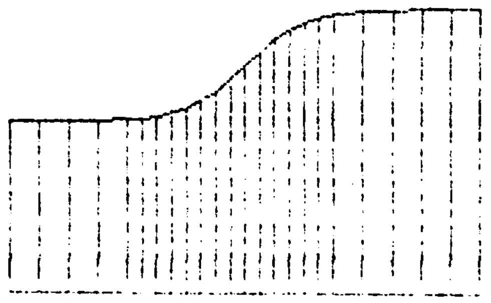
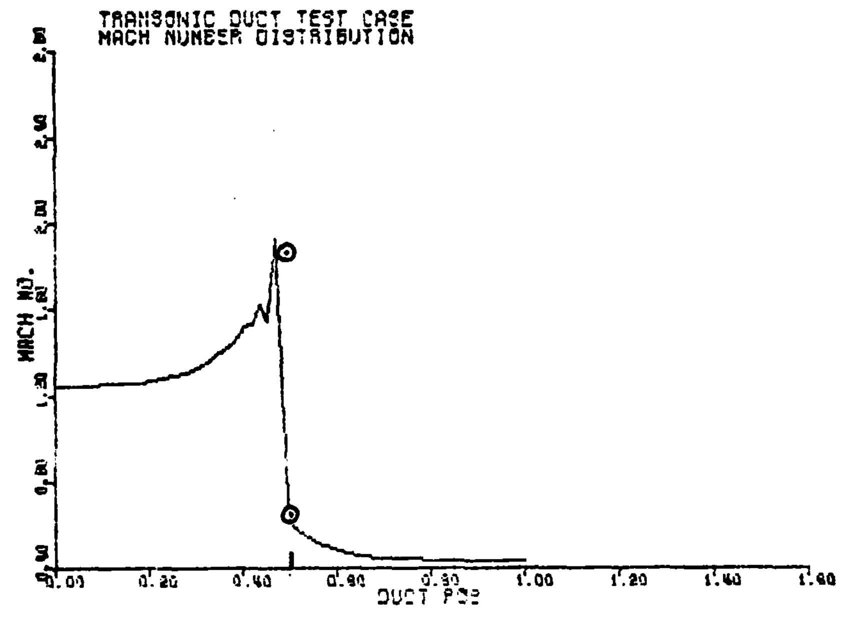


Figure 3 Mach number distribution for divergent duct flow with supersonic inlet/subsonic exhaust conditions

Level 1 Adaptation to shock feature



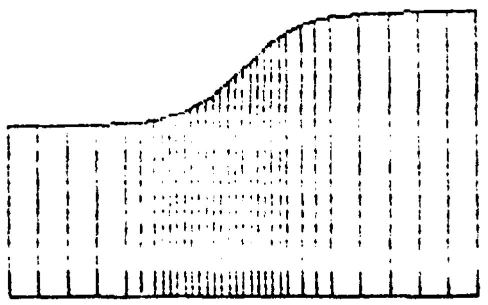
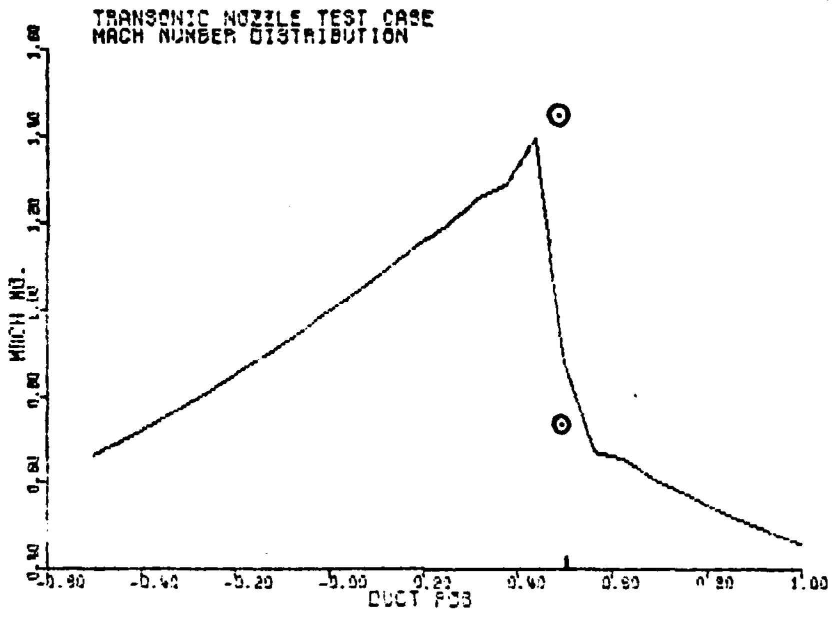


Figure 4 Mach number distribution for divergent duct flow with supersonic inlet/subsonic exhaust conditions

Level 2 and 1 Adaptations to shock feature



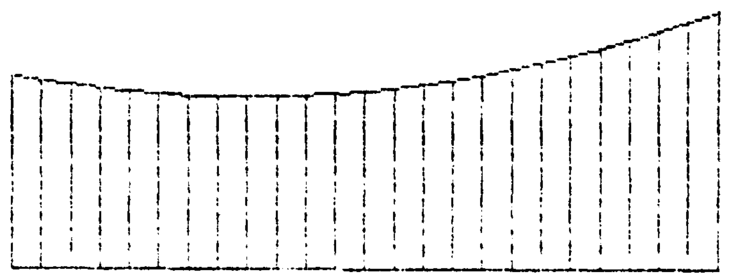
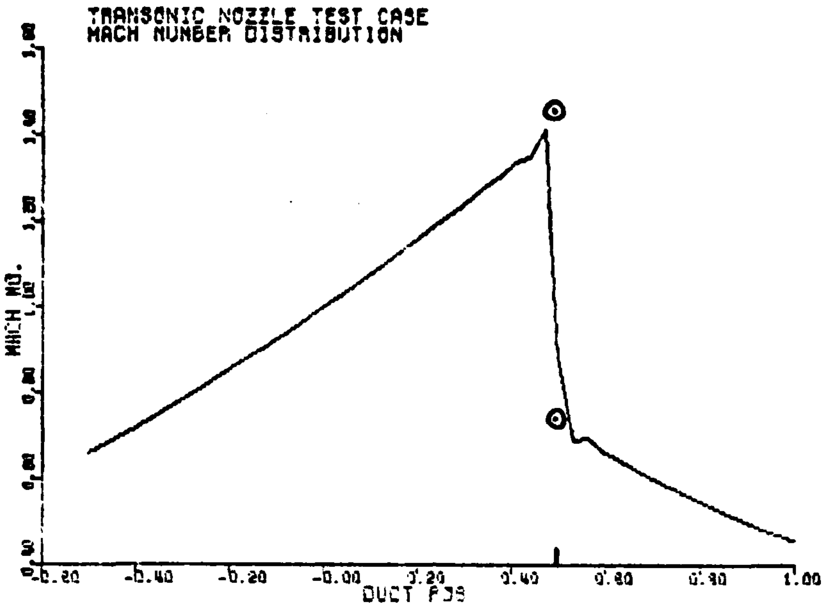


Figure 5 Mach number distribution for converging diverging nozzle with subsonic inlet and exhaust and intervening shock

Level 0, Uniform global grid



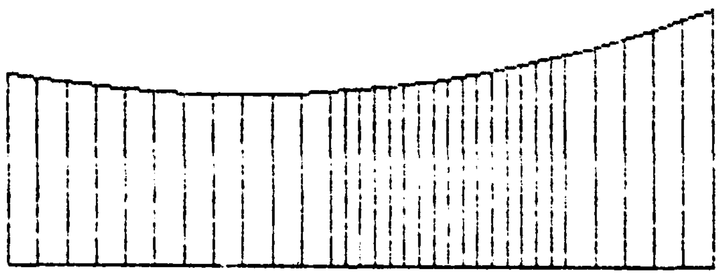
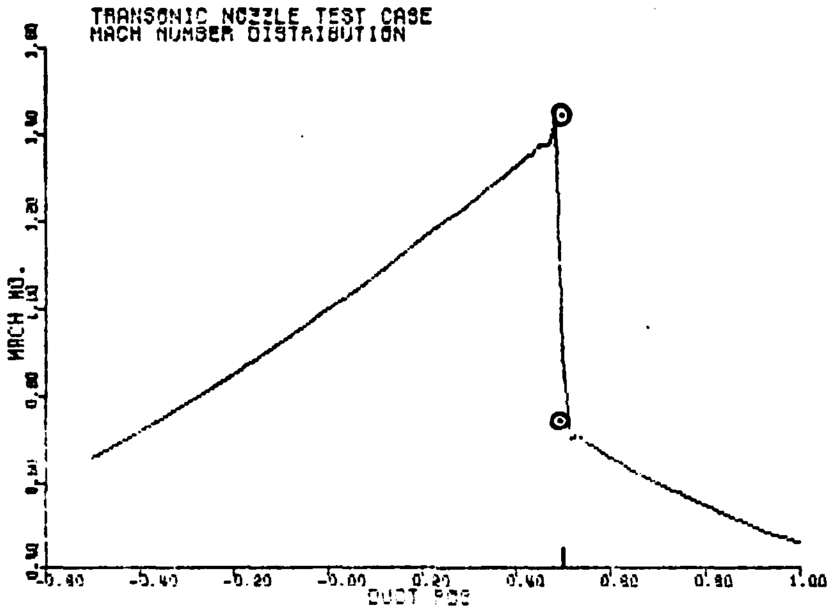


Figure 6 Mach number distribution for converging diverging nozzle with subsonic inlet and exhaust and intervening shock

Level 1 Adaptation to shock feature



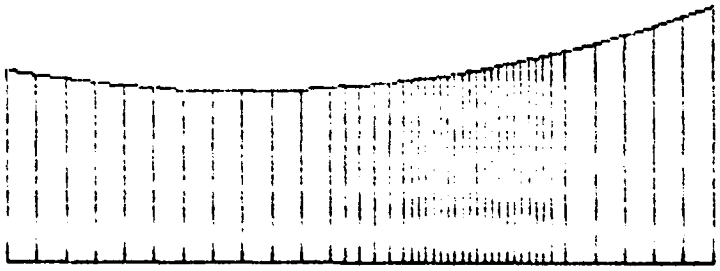
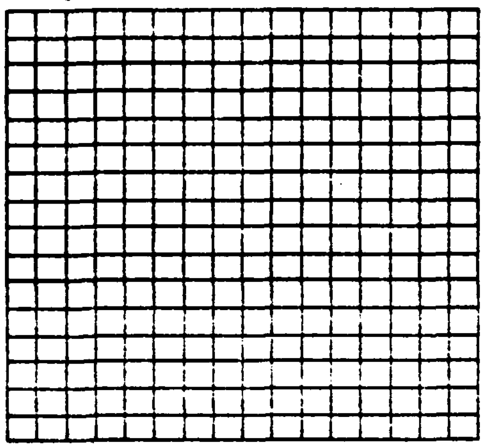
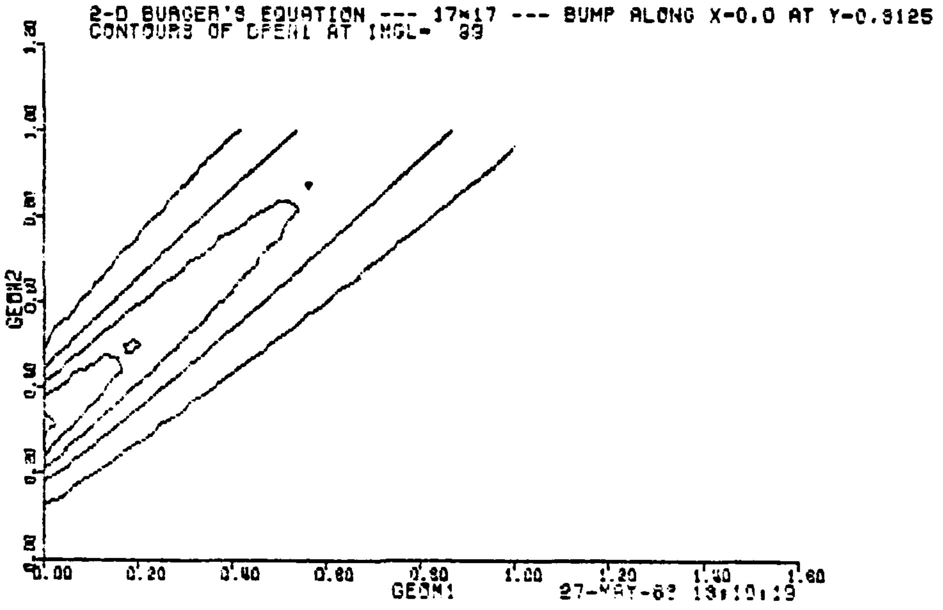


Figure 7 Mach number distribution for converging diverging nozzle with subsonic inlet and exhaust and intervening shock

Level 2 and 1 Adaptations to shock feature




2-D BURGER'S EQUATION --- 17×17 --- BUMP ALONG X=0.0 AT Y=0.3125
CONTOURS OF DFEN1 AT IMGL= 89
GEOM2
GEOM1
27-MAY-82 13:10:19

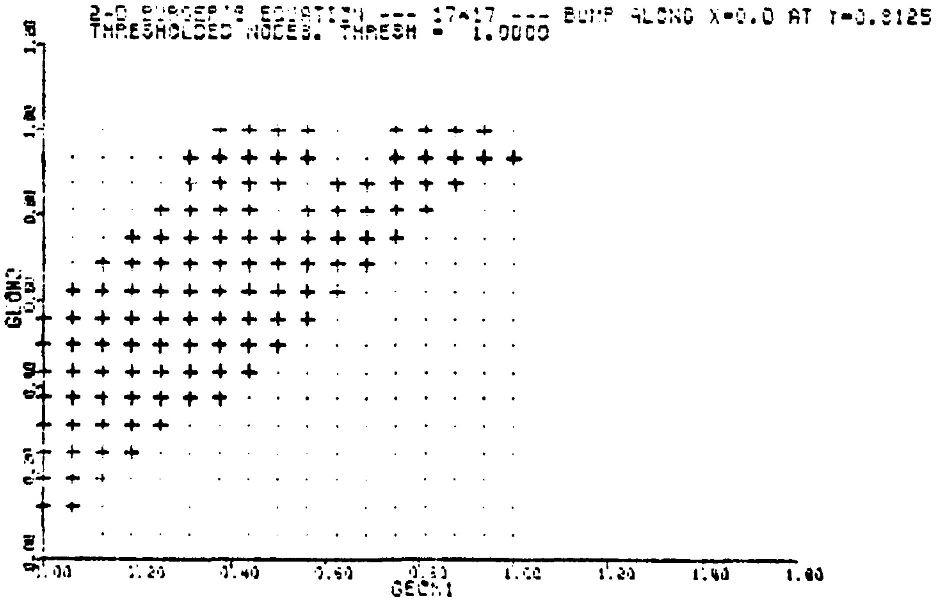
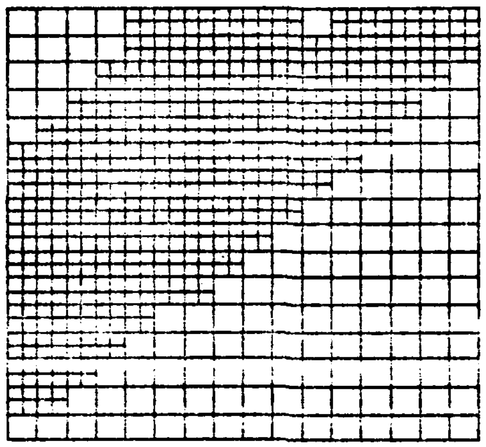


Figure 8 Two-dimensional field contours for model convection/diffusion equation (4)

Above: Level 0, Uniform global grid basis

Below: [illegible]



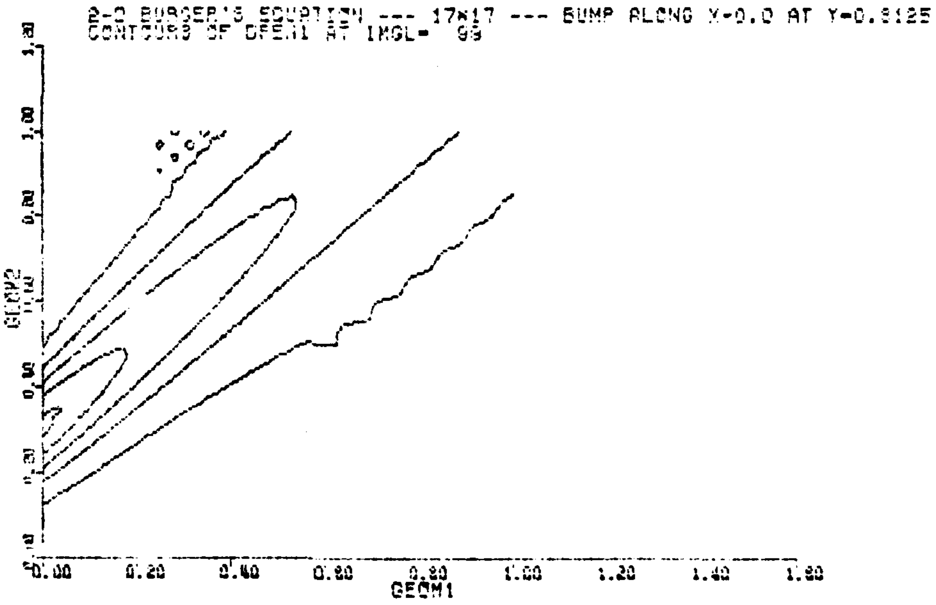

2-D BURGER'S EQUATION --- 17X17 --- BUMP ALONG X=0.0 AT Y=0.3125
CONTOURS OF DFEN1 AT IMGL= 99
GEOM2
GEOM1
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60

Figure 9 Two-dimensional field contours for model convection/diffusion equation (4)

Above: Level 1 Adaptation to disturbed field basis

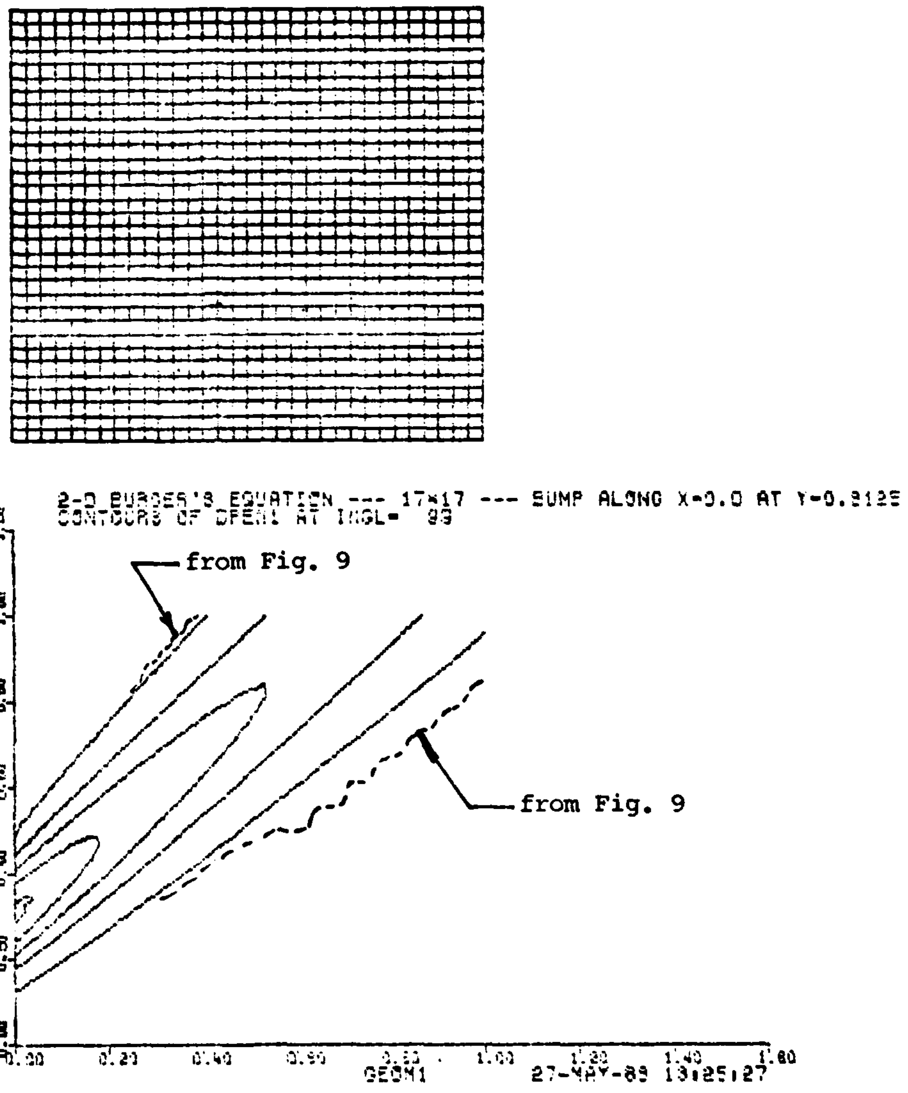


Figure 10 Two-dimensional field contours for model convection/diffusion equation (4)

Above: Level 1, Uniform grid basis

DAT
FILM